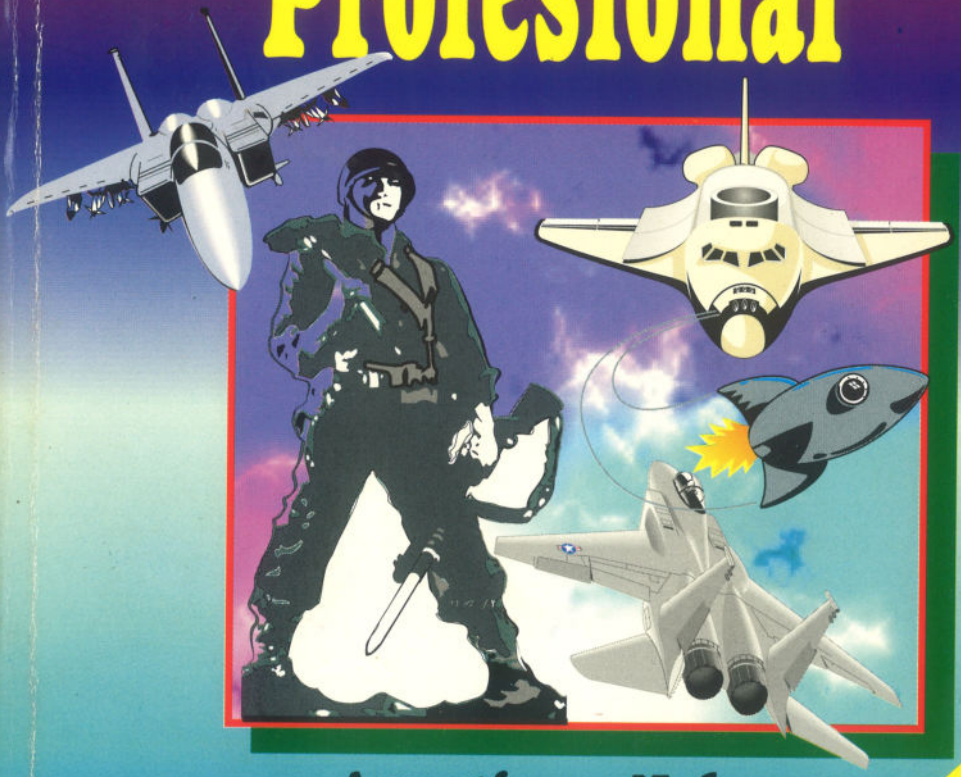
ELEX MEDIA KOMPUTINDO

SERI APLIKASI PEMROGRAMAN

Pemrograman Animasi dan Game Profesional



Agustinus Nalwan

Disertai DISKET

SERI APLIKASI PEMROGRAMAN

Pemrograman Animasi dan Game Profesional

Agustinus Nalwan

Seri Aplikasi Pemrograman

Pemrograman Animasi dan Game Profesional

Sanksi Pelanggaran Pasal 44:
Undang-undang Nomor 7 Tahun 1987 Tentang Perubahan atas undang-undang Nomor 6 Tahun 1982 Tentang Hak Cipta

1. Barangsiapa dengan sengaja dan tanpa hak mengumumkan atau memperbanyak suatu ciptaan atau memberi izin untuk itu, dipidana dengan pidana penjara paling lama 7 (tujuh) tahun dan/atau denda paling banyak Rp 100.000.000,- (seratus juta rupiah).
2. Barangsiapa dengan sengaja menyiarkan, memamerkan, mengedarkan, atau menjual kepada umum suatu ciptaan atau barang hasil pelanggaran Hak Cipta sebagaimana dimaksud dalam ayat (1), dipidana dengan pidana penjara paling lama 5 (lima) tahun dan/atau denda paling banyak Rp 50.000.000,- (lima puluh juta rupiah).

Seri Aplikasi Pemrograman

Pemrograman Animasi dan Game Profesional

Agustinus Nalwan

Penerbit PT Elex Media Komputindo
Kelompok Gramedia - Jakarta

Seri Aplikasi Pemrograman
Pemrograman Animasi dan Game Profesional
Agustinus Nalwan
12195321
ISBN 979-637-340-8
Sampul: Slamet M. Jaeni
Perwajahan: Yudi S. & PL Aristiawati
©1995,PT Elex Media Komputindo, Jakarta
Hak Cipta dilindungi Undang-undang
Diterbitkan pertama kali oleh
Penerbit PT Elex Media Komputindo
Kelompok Gramedia, Anggota IKAPI, Jakarta 1995

Cetakan pertama : Mei 1995
Cetakan kedua : Januari 1996

Dilarang keras menterjemahkan, memfotokopi, atau memperbanyak sebagian atau seluruh isi buku ini tanpa izin tertulis dari penerbit.

Dicetak oleh Percetakan PT Gramedia, Jakarta
Isi di luar tanggung jawab percetakan

PENDAHULUAN

Buku ini merupakan dasar dari pemrograman animasi dan game profesional. Dengan buku ini Anda akan diajak untuk mempelajari dasar-dasar pembuatan game profesional yang beredar di pasaran menggunakan compiler Turbo Pascal 7.00 yang dibantu oleh routine-routine Assembler.

Pada bab-bab awal akan dibahas tentang penggunaan GRAPH.TPU dari Turbo Pascal untuk memperkenalkan Anda tentang animasi. Selanjutnya kita akan merakit unit graphic sendiri yaitu VGACAD.TPU. dengan unit ini Anda telah dapat merakit program game Anda sendiri yang setaraf dengan game-game yang beredar di pasaran seperti RAPTOR, STREET FIGHTER, dan sebagainya. pada bab yang terakhir, kita akan mempraktekkan apa yang telah kita pelajari dengan membuat game bertipe RAPTOR.

DISKET PENUNJANG BUKU

Disket yang menyertai buku ini berisi contoh-contoh program yang ada dalam buku. Untuk menginstalnya, masukkan disket tersebut ke dalam disk driver kemudian ketikkan:

install drive_asal drive_tujuan [tekan Enter]

Misalnya, jika ingin menginstal dari drive A (disket berada pada drive A) ke drive C (harddisk), ketikkan **install A: C:** [tekan Enter]. Anda tidak perlu menuliskan direktori tujuan karena program install akan membuatkannya jika Anda menjawab [Y]a pada pertanyaan yang diberikan. Dengan perintah ini, program install akan mendekompresi file-file terkompres yang ada dalam disket.

Hasil instal akan membuat direktori **gamesprog** dan tiga sub direktori **exe, fire**, dan **pas** yang dapat menunjang pemahaman Anda dalam mempelajari buku ini.

Selamat mencoba!

DAFTAR ISI

PENDAHULUAN v

Bab 1 **Dasar Pemrograman Animasi dan Game I** 1
Pendahuluan 2
Pengenalan Unit Turbo Pascal GRAPH.TPU 2
Animasi Dasar dengan GRAPH.TPU 3
Pemrograman dengan GRAPH.TPU 3
Fasilitas-Fasilitas GRAPH.TPU 7
Animasi dengan GRAPH.TPU 13
Pembuatan Game Sederhana dengan GRAPH.TPU 18

Bab 2 **Dasar Pemrograman Animasi dan Game II** 31
Pendahuluan 32
Kelemahan-Kelemahan GRAPH.TPU 32
Penggunaan Mode Graphic 13h 41
Alokasi Pixel pada Memori Layar 42
Pendahuluan 42
Alokasi Pixel secara Langsung 42
Pembuatan Game dengan Alokasi Memori Layar 46
Aplikasi Palette pada Monitor VGA 60

Bab 3 **Pembuatan Unit VGACAD.TPU** 67
Pengenalan Unit VGACAD.TPU 68
Instalasi Mode Graphic 13h 83
Pembuatan Bangun-Bangun Geometri 84
Fasilitas-Fasilitas Pembuatan Bangun Geometri 84
Praktek Pembuatan Bangun Geometri dengan VGACAD.TPU 95
Animasi Tiga Layar 98
Fasilitas-Fasilitas Animasi Tiga Layar 107
Praktek Animasi Tiga Layar dengan VGACAD.TPU 120

Pengolahan Gambar 131
Aplikasi Animasi Palette 141
Fasilitas Manipulasi Palette pada VGACAD.TPU 142
Pemanfaatan Fasilitas Palette pada VGACAD.TPU 151
Animasi Scroll 164
Fasilitas Scroll Page pada VGACAD.TPU 167
Pemanfaatan Fasilitas Scroll Page pada VGACAD.TPU 172

Bab 4 Pembuatan Game Profesional 181
Pendahuluan 182
Pembuatan Proyek Game Profesional 183
Perencanaan 183
Pembuatan Gambar 184
Pembuatan Editor 184
Pembuatan Engine 185
Pembuatan Data & Level 185
Pembenahan Akhir 185

Bab 5 Pembuatan Game FIRE FIGHTER 187
Perencanaan 188
Pembuatan Gambar 191
Fire Fighter Mission Editor 192
Pendahuluan 192
Pemakaian Fire Fighter Mission Editor 194
Pembuatan Fire Fighter Mission Editor 204
Fire Fighter Game Engine 264
Pendahuluan 264
Pembuatan Fire Fighter Game Engine 264

Informasi Daftar File 313
Penutup 317

BAB 1

DASAR PEMROGRAMAN ANIMASI DAN GAME I

PENDAHULUAN

Definisi dari animasi adalah gambar yang bergerak, contohnya seperti iklan-iklan di televisi, film kartun, game dan sebagainya. Bahkan lampu berjalan di papan advertising pun juga merupakan animasi.
Animasi adalah bagian yang sangat menarik untuk dipelajari, terutama bagi Anda yang tertarik pada pemrograman graphic dan game. Perlu Anda ketahui, game juga merupakan animasi yang dapat dikontrol oleh pemain.

Memang buku ini akan membahas pemrograman game profesional, tetapi bagi Anda yang masih pemula janganlah merasa khawatir karena pada dua bab awal ini kita akan mempelajari pemrograman animasi mulai dari dasarnya.

PENGENALAN UNIT TURBO PASCAL GRAPH.TPU

Tentu sebagian besar dari Anda pernah menggunakan GRAPH.TPU dalam pemrograman graphic. GRAPH.TPU ini adalah merupakan unit dari Turbo Pascal yang menyediakan banyak fasilitas untuk pemrograman graphic. Procedure-procedure untuk pembuatan garis, lingkaran, pengisian warna dan sebagainya ada di dalam unit ini.

Dalam penggunaanya, GRAPH.TPU perlu dibantu dengan driver-driver yang dikemas dalam file berekstensi BGI. Di antaranya ialah CGA.BGI dan EGAVGA.BGI. BGI adalah singkatan dari "Borland Graphic Interface". Kegunaan dari file BGI ini adalah sebagai driver monitor yang Anda gunakan. CGA.BGI digunakan untuk monitor CGA sedangkan EGAVGA.BGI digunakan untuk monitor EGA dan VGA.

Di samping file-file BGI di atas, terdapat pula file-file font berekstensi OBJ yang menyimpan tipe-tipe huruf yang dapat Anda gunakan, seperti BOLD.OBJ dan GOTHIC.OBJ. Dengan adanya file-file ini, Anda dapat menampilkan tulisan dalam mode graphic dengan model/style yang berbeda-beda.

PEMROGRAMAN DENGAN GRAPH.TPU

Untuk mempergunakan unit ini, Anda harus melakukan instalasi pada awal program dengan menyertakan tipe monitor dan mode graphic yang Anda kehendaki. Adapun sebagian besar dari mode graphic dan tipe monitor yang dapat Anda pilih terdapat pada tabel berikut.

NO	TIPE MONITOR	MODE GRAPHIC	RESOLUSI	JUMLAH WARNA
1	CGA	CGAC0	320 x 200	16
		CGAC	320 x 200	16
		CGAC2	320 x 200	16
		CGAC3	320 x 200	16
		CGAHi	640 x 200	16
2	EGA	EGALo	640 x 200	16
		EGAHi	640 x 350	16
3	VGA	VGALo	640 x 200	16
		VGAMed	640 x 350	16
		VGAHi	640 x 480	16

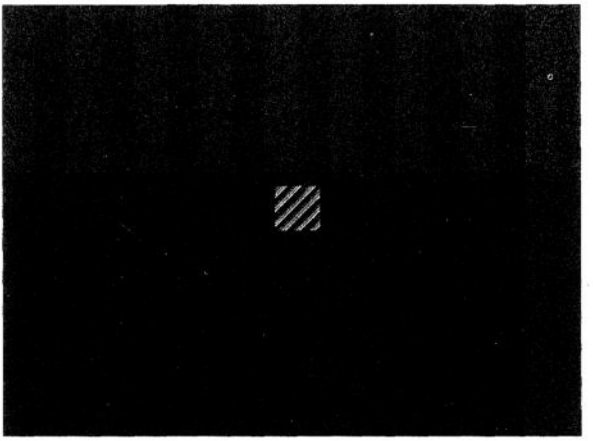
Sekarang coba perhatikan program di bawah ini.

Listing Program

```
{ ------------------------------------------------------------ }
{ File     : Anim1.Pas                                         }
{ Compiler : Turbo Pascal 7.0                                  }
{ Tujuan   : Menginstal Mode Graphic VGA 640 x 480 16 color    }
{ ------------------------------------------------------------ }
Uses Crt,Graph,Dos;
Var X,Y                 : Integer;
      Gr_Mode,Sc_Mode   : Integer;
      Ch                : Char;
```

```
Procedure Initialize;
  Begin
    Gr_Mode:= Vga;
    Sc_Mode:= VgaHi;
    InitGraph(Gr_Mode,Sc_Mode,'' ");
  End;

Procedure Main;
  Begin
    For X:=1 to 50 do
      For Y:=1 to 50 do
        PutPixel(300+X,200+Y,(X+Y) mod 16);
        Ch:=Readkey;
  End;
Begin
  Initialize;
  Main;
  CloseGraph;
End.
```

Program ini adalah program untuk menampilkan balok berukuran 50 x 50 pixel pada tengah-tengah layar dengan warna yang beraneka ragam.



Gambar 1.1 Output Screen dari program ANIM1.PAS.

Pada awal program dipanggil procedure INITIALIZE. Procedure ini berguna untuk melakukan installasi mode graphic.

```
Procedure Initialize;
  Begin
    Gr_Mode:= Vga;
    Sc_Mode:= VgaHi;
    InitGraph(Gr_Mode,Sc_Mode,'' ");
  End;
```

Statement : INITGRAPH.
Fungsi : Menginstall layar menjadi mode graphic.
Sintaks : InitGraph(Tipe_Monitor,Graphic_Mode,Directory);
Parameter :

Tipe_Monitor	=	Variabel byte Menentukan tipe monitor yang Anda gunakan.
Graphic_Mode	=	Variabel byte Menentukan Graphic mode yang Anda inginkan.
Directory	=	Variabel String Menunjukkan direktori dimana GRAPH.TPU berada.

Pada procedure di atas, graphic diinstall dengan tipe monitor VGA dan mode graphic yang diinginkan adalah VgaHi yaitu dengan resolusi 640 x 400 pixel 16 warna. Sedangkan pada procedure Main, dilakukan proses pembuatan balok dengan cara peletakan titik secara berulang ke kanan dan ke bawah. Hasilnya ialah layar koordinat (301,201) hingga (350,250) akan diblok dengan 16 warna yang berbeda. Untuk peletakan titik, digunakan statement PUTPIXEL.

```
Procedure Main;
  Begin
    For X:=1 to 50 do                    {Pengulangan Horizontal}
      For Y:=1 to 50 do                  {Pengulangan Vertical}
        PutPixel(300+X,200+Y,(X+Y) mod 16); {Taruh titik}
    Ch:=Readkey;                         {Tunggu penekanan tombol}
  End;
```

Statement : PUTPIXEL.
Fungsi : Untuk meletakkan titik di layar.
Sintaks : Putpixel(X,Y,Col);
Parameter :

X = Variabel Integer
Untuk menentukan koordinat X dari titik.

Y = Variabel Integer
Untuk menentukan koordinat Y dari titik.

Col = Variabel Byte
Untuk menentukan warna dari titik.

Batas maksimum koordinat X, Y dan jumlah warna yang dapat dipilih bergantung pada mode graphic yang Anda pilih. Apabila resolusi yang Anda gunakan adalah 640 x 480, maka batas maksimum koordinat X adalah 639 (dimulai dari X = 0) sedangkan koordinat Y adalah 479 (dimulai dari Y = 0). Sedangkan warna yang dapat Anda gunakan juga bergantung pada resolusi monitor. Parameter Col dapat diisi dengan nomor kode warna yang Anda inginkan atau memakai konstanta yang telah disediakan oleh GRAPH.TPU. Perhatikan tabel berikut.

NO	WARNA	KONSTANTA	CODE
0	Hitam	Black	0
1	Biru	Blue	1
2	Hijau	Green	2
3	Ungu	Cyan	3
4	Merah	Red	4
5	Magenta	Magenta	5
6	Coklat	Brown	6
7	Abu-abu terang	LightGray	7
8	Abu-abu gelap	DarkGray	8
9	Biru tua	LightBlue	9
10	Hijau tua	LightGreen	10
11	Ungu tua	LightCyan	11
12	Merah tua	LightRed	12
13	Magenta tua	Light Magenta	13
14	Kuning	Yellow	14
15	Putih	White	15

Sedangkan CloseGraph digunakan untuk menutup layar mode graphic dan kembali ke mode text.

```
   CloseGraph;
End.
```

Statement : CLOSEGRAPH.
Fungsi : Untuk menutup mode graphic.
Sintaks : CloseGraph;

Tentunya dari program sederhana di atas Anda telah mendapat gambaran tentang GRAPH.TPU dari Turbo Pascal. Selain statement PUTPIXEL di atas, masih banyak statement lain yang disediakan. Untuk itu sekarang kita akan mempelajari beberapa diantaranya. Bila Anda ingin penjelasan statement secara lengkap, Anda dapat membacanya di buku USER GUIDE dari Turbo Pascal karena kita tidak akan berlama-lama menggunakan GRAPH.TPU. Setelah Bab I selesai kita akan meninggalkan GRAPH.TPU ini dan mulai dengan pemrograman animasi dan game profesional.

FASILITAS-FASILITAS GRAPH.TPU

Statement: SETCOLOR.
Fungsi : Untuk menentukan warna yang aktif.
Sintaks : SetColor(Col);
Parameter:
Col = Variabel Byte
Untuk menentukan warna yang aktif.

Statement: GETCOLOR.
Fungsi : Untuk mengetahui warna yang sedang aktif.
Sintaks : GetColor(Col);
Parameter:
Col = Variabel Byte
Untuk menerima nomor warna yang sedang aktif.

Statement: CIRCLE.
Fungsi : Untuk membuat lingkaran di layar.
Sintaks : Circle(X,Y,Rad);
Parameter:
X = Variabel Integer
Untuk menentukan koordinat X dari titik pusat lingkaran.

Y = Variabel Integer
Untuk menentukan koordinat Y dari titik pusat lingkaran.

Rad = Variabel Integer
Untuk menentukan Jari-jari dari lingkaran.

Statement: LINE.
Fungsi : Untuk membuat garis di layar.
Sintaks : Line(X1,Y1,X2,Y2);
Parameter:

X1 = Variabel Integer
Koordinat X dari titik asal.

Y1 = Variabel Integer
Koordinat Y dari titik asal.

X2 = Variabel Integer
Koordinat X dari titik tujuan.

Y2 = Variabel Integer
Koordinat Y dari titik tujuan.

Statement: RECTANGLE.
Fungsi : Untuk membuat persegi panjang di layar.
Sintaks : Rectangle(X1,Y1,X2,Y2);
Parameter:

X1 = Variabel Integer
Koordinat X dari titik pertama.

Y1 = Variabel Integer
Koordinat Y dari titik pertama.

X2 = Variabel Integer
Koordinat X dari titik kedua.

Y2 = Variabel Integer
Koordinat Y dari titik kedua.

Statement: SETFILLSTYLE.
Fungsi : Untuk menentukan warna dan pola arsiran pada statement FLOODFILL.
Sintaks : SetFillStyle(Pola_Arsir,Col);
Parameter:

Pola_Arsir = Variabel Byte
Untuk menentukan pola arsiran

Adapun pola-pola arsiran itu dapat Anda lihat pada tabel berikut.

NO	CONSTANTA	CODE	POLA ARSIRAN
0	EmptyFill	0	
1	SolidFill	1	
2	LineFill	2	
3	LtSlashFill	3	
4	SlashFill	4	
5	BkSlashFill	5	
6	LtBkSlashFill	6	
7	HatchFill	7	
8	XHatchFill	8	
9	InterLeaveFill	9	
10	WideDotFill	10	
11	CloseDotFill	11	
12	UserPattern	12	

Col = Variabel Byte
Untuk menentukan warna arsiran.

Statement: FLOODFILL.
Fungsi : Untuk mengisi suatu kurva tertutup dengan warna dan pola tertentu.
Sintaks : FloodFill(X,YCol);
Parameter:

X = Variabel Integer
Untuk menentukan koordinat X dari titik di dalam kurva tertutup yang akan diisi warna.

Y = Variabel Integer
Untuk menentukan koordinat Y dari titik di dalam kurva tertutup yang akan diisi warna.

Col = Variabel Byte
Untuk menentukan warna kurva tertutup.

Statement: CLEARDEVICE
Fungsi : Untuk menghapus layar.
Sintaks : ClearDevice;
Statement: GETIMAGE.

Fungsi : Untuk merekam suatu area di layar monitor ke dalam memori.

Sintaks : GetImage(X1,Y1,X2,Y2,Sprite^);

Parameter:

X1 = Variabel Integer.
Koordinat X dari pojok kiri atas area.

Y1 = Variabel Integer.
Koordinat Y dari pojok kiri atas area.

X2 = Variabel Integer.
Koordinat X dari pojok kanan bawah area.

Y2 = Variabel Integer.
Koordinat Y dari pojok kanan bawah area.

Sprite = Variabel Pointer.
Pointer yang menunjukkan alamat di mana gambar disimpan sehingga variabel Sprite harus dideklarasikan dahulu dengan fungsi GETMEM.

Statement: IMAGESIZE.

Fungsi : Untuk mengetahui memori yang diperlukan untuk menyimpan gambar di memori.

Sintaks : Size:=ImageSize(X1,Y1,X2,Y2);

Parameter:

Size = Variabel Word.
Menerima angka yang menunjukkan besar memori yang dibutuhkan untuk merekam gambar dengan GETIMAGE.

X1 = Variabel Integer.
Koordinat X dari pojok kiri atas area.

Y1 = Variabel Integer.
Koordinat Y dari pojok kiri atas area.

X2 = Variabel Integer.
Koordinat X dari pojok kanan bawah area.

Y2 = Variabel Integer.
Koordinat Y dari pojok kanan bawah area.

Statement: PUTIMAGE.

Fungsi : Untuk menampilkan gambar yang disimpan dalam memori ke layar.

Sintaks : PutImage(X1,Y1,Sprite^,Tipe);

Parameter:

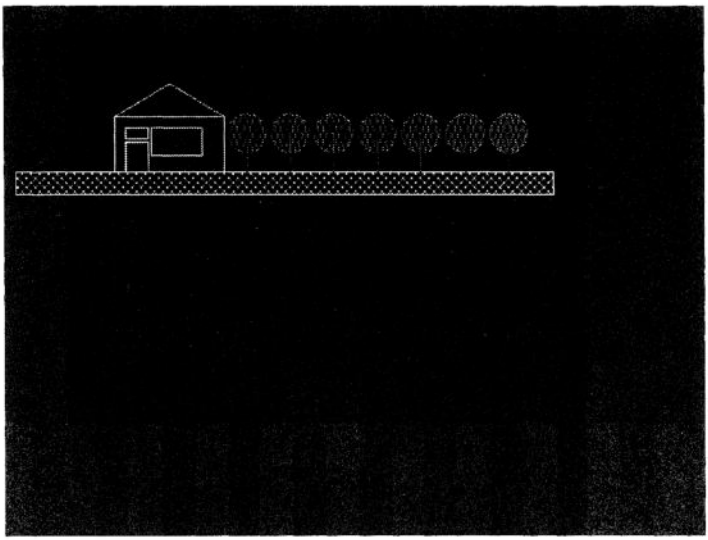
X1	= Variabel Integer. Koordinat X dimana gambar akan diletakkan di layar.
Y1	= Variabel Integer. Koordinat Y dimana gambar akan diletakkan di layar. Pointer yang menunjukkan alamat di mana gambar berada. Untuk itu Sprite harus telah berisi gambar yang direkam melalui GETIMAGE.
Tipe	= Variabel Byte. Menunjukkan tipe penulisan gambar ke layar. Adapun tipe-tipe penulisan gambar ke layar tersebut adalah XORPUT, NOTPUT, NORMALPUT dan ANDPUT.

Program di bawah ini sangat bagus untuk Anda dalam mempelajari pemakaian dari statement-statement di atas. Setelah program ini dijalankan, pada layar monitor akan digambar denah rumah beserta halamannya secara sederhana. Semuanya dibuat dengan menggunakan statement-statement di atas, cobalah untuk mempelajarinya.

Listing Program

```
Program Graph_Tpu2;
{ ------------------------------------------------------------ }
{ File     : Anim2.Pas                                         }
{ Compiler : Turbo Pascal 7.0                                  }
{ Tujuan   : Menggambar Rumah dengan Fungsi GRAPH.TPU          }
{  ------------------------------------------------------------}

Uses Crt,Graph,Dos;

Var X,Y             : Integer;
      Gr_Mode,Sc_Mode: Integer;
      Ch             : Char;

Procedure Initialize;
  Begin
    Gr_Mode:= Vga;
```

```
    Sc_Mode:= VgaHi;
    InitGraph(Gr_Mode,Sc_Mode,'');
  End;

Procedure Main;
  Begin
    SetColor(15);
    Rectangle(10,150,500,170);          {Pembuatan Pondasi       }
    Rectangle(100,100,200,150);         {Pembuatan Tembok        }
    Line(100,100,150,70);               {Pembuatan Atap          }
    Line(150,70,200,100);               {Pembuatan Atap          }
    Rectangle(110,110,130,120);         {Pembuatan Jendela Kecil}
    Rectangle(110,123,130,150);         {Pembuatan Pintu         }
    Rectangle(133,110,180,135);         {Pembuatan Jendela Besar}
    SetFillStyle(7,Red);
    FloodFill(150,73,15);               {Pewarnaan Atap          }
    SetFillStyle(1,Brown);
    FloodFill(101,101,15);              {Pewarnaan Dinding       }
    SetFillStyle(1,LightBlue);
    FloodFill(111,111,15);              {Pewarnaan Jendela Kecil}
    FloodFill(134,111,15);              {Pewarnaan Jendela Besar}
    SetFillStyle(1,DarkGray);
    FloodFill(111,124,15);              {Pewarnaan Pintu         }
    SetFillStyle(8,Yellow);
    FloodFill(11,151,15);               {Pewarnaan Pondasi       }
    SetFillStyle(9,LightGreen);
    For X:=0 to 6 do
      Begin
        SetColor(Green);
        Circle(220+X*40,115,18);        {Pembuatan Pohon         }
        Floodfill(220+X*40,114,Green); {Pewarnaan Pohon         }
        SetColor(Brown);
        Line(220+X*40,115,220+X*40,150); {Pembuatan Batang Pohon}
      End;
  Ch:=Readkey;
End;

Begin
  Initialize;
  Main;
  CloseGraph;
End.
```



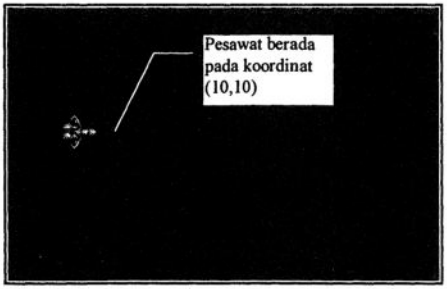
Gambar 1.2 Output Screen dari program ANIM2.PAS.

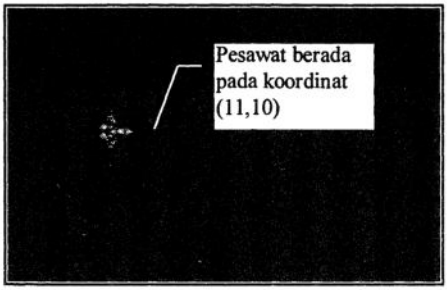
ANIMASI DENGAN GRAPH.TPU

Apabila Anda belum pernah memprogram animasi tentu di benak Anda timbul pertanyaan bagaimana sih cara pembuatan gambar bergerak di layar? Tentunya Anda sekalian pernah melihat game di TV Game baik yang di dink-donk maupun yang di rumah seperti Super Nintendo, Sega dan sebangsanya. Bagaimana proses pembuatan pesawat tempur yang bisa dikontrol ke kanan, kiri, atas, bawah. Bagaimana cara membuat gerakan orang memukul, menendang, lompat seperti di STREET FIGHTER dan DOUBLE DRAGON.

Sebenarnya gambar pesawat atau orang yang bergerak tersebut ialah gambar yang diletakkan pada posisi yang berbeda-beda setiap waktu dan karena cepatnya, kita melihat gambar tersebut seakan-akan bergerak. Kita ambil contoh gambar pesawat yang bergerak ke kanan.

Mula-mula gambar pesawat tersebut diletakkan di koordinat (10,10) kemudian dengan cepat gambar tersebut dihapus dan gambar pesawat diletakkan lagi di koordinat (11,10) lalu dihapus lagi dan diletakkan lagi di koordinat (12,10). Demikian seterusnya sehingga seakan-akan kita melihat pesawat tersebut bergeser ke kanan layar.

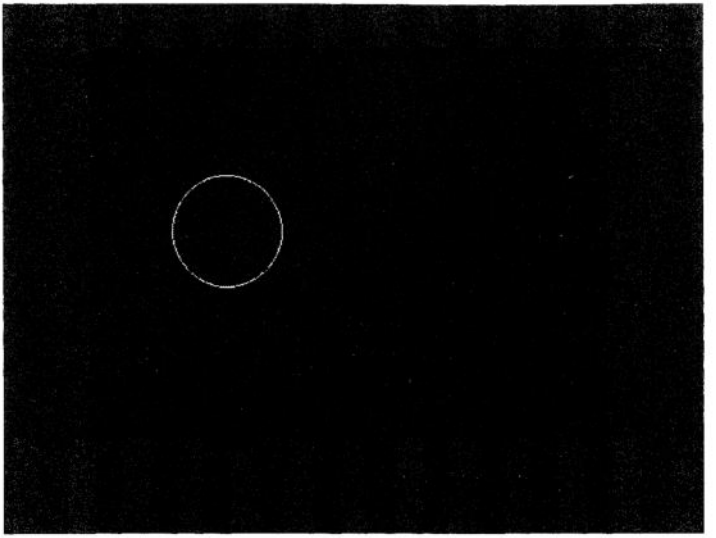




Untuk membuat pesawat tersebut bergerak ke kanan bawah layar, selain menambah koordinat X, kita juga menambah koordinat Y. Misalnya posisi awal pesawat di koordinat (10,10) dan hendak menuju ke koordinat (15,15). Pada setiap frame dilakukan proses penambahan koordinat X dan Y hingga nilai X dan Y sudah mencapai 15. Untuk memberikan gambaran pada Anda, coba program berikut.

```
Program Graph_Tpu3;
{ ------------------------------------------------------------ }
{ File     :  Anim3.Pas                                        }
```

```
{ Compiler :  Turbo Pascal 7.0                                          }
{ Tujuan   :  Menggerakkan Bola Sebagai Sprite Ke Kanan Layar           }
{  ------------------------------------------------------------------}

Uses Crt,Graph,Dos;

Var X,Y             : Integer;
      Gr_Mode,Sc_Mode: Integer;
      Ch             : Char;
      Sprite         : Pointer;

Procedure Initialize;
  Begin
    Gr_Mode:= Vga;
    Sc_Mode:= VgaHi;
    InitGraph(Gr_Mode,Sc_Mode,'');
  End;

Procedure Draw_Sprite;
  Begin
    SetColor(15);
    Circle(60,60,50);
    SetFillStyle(1,8);
    FloodFill(60,60,15);
  End;

Procedure Store_Sprite;
  Begin
    GetMem(Sprite,ImageSize(10,10,110,110));
    GetImage(10,10,110,110,Sprite^);
  End;

Procedure Main;
  Begin
    ClearDevice;
    For X:=1 to 500 do
      Begin
        PutImage(X,X,Sprite^,NormalPut);
        Delay(10);
        PutImage(X,X,Sprite^,XorPut);
     End;
 End;

Begin
  Initialize;
  Draw_Sprite;
  Store_Sprite;
  Main;
  CloseGraph;
End.
```



Gambar 1.3 Output Screen dari program ANIM3.PAS.

Penjelasan Listing Program

Program ini akan menampilkan gambar lingkaran berarsir pada layar monitor. Setelah itu layar akan dibersihkan dan Anda akan melihat bola tadi bergerak dari kiri atas layar ke kanan bawah layar. Seperti biasa, procedure INITIALIZE berguna untuk meset layar dengan mode graphic. Mode graphic yang dipilih adalah VGAHI (640 x 480 16 colors).
Kemudian dipanggil procedure DRAW_SPRITE yang berisi sejumlah perintah untuk menggambar lingkaran berarsir di layar monitor.

```
Procedure Draw_Sprite;
  Begin
    SetColor(15);       {Set warna yang aktif menjadi putih      }
    Circle(60,60,50);  {Menggambar lingkaran di 60,60 radius 50}
    SetFillStyle(1,8); {Mengisi lingkaran dengan warna 8 (abu- }
    FloodFill(60,60,15);{abu) dengan model arsiran nomor 1     }
  End;
```

Langkah berikutnya memanggil procedure STORE_SPRITE. Disini dilakukan perekaman gambar dari layar ke memori dengan statement GETIMAGE. Alamat pointer memori yang dituju adalah SPRITE. Untuk

itu dilakukan pemesanan memori kepada TURBO PASCAL dengan statement GETMEM sebesar jumlah byte yang dibutuhkan untuk menyimpan gambar tersebut. Adapun rumus besar memori yang dibutuhkan untuk menyimpan gambar di dalam kotak koordinat kiri atas (X1,Y1) dan kanan bawah (X2,Y2) adalah sebagai berikut:

$$Size=((X2-X1+1)*(Y2-Y1+1))/2+4$$

Rumus ini diperoleh dari rumus luas persegi panjang. Tentu kita tahu bahwa untuk menyimpan data sejumlah Nx ke kanan dan Ny ke bawah adalah sama dengan menghitung luas persegi panjang Nx dikali dengan Ny. Pembagian dengan dua pada perhitungan (X2-X1+1) dan (Y2-Y1+1) di atas dilakukan karena di dalam satu byte alamat memori kita dapat menyimpan nilai dalam range 0 sampai 255 = 8 bit. Sedangkan pada satu pixel di mode layar ini, warna yang dapat ditampilkan berjumlah 16 = 4 bit (Mode graphic yang dipergunakan adalah 640 x 480 16 warna) sehingga untuk menyimpan warna dari tiap titik, cukup diperlukan variabel 4 bit. Untuk itulah GRAPH.TPU membuat system penyimpanan dimana satu alamat memori menyimpan dua titik sekaligus. Sedangkan 4 byte ditambahkan karena kita juga perlu merekam dua buah data INTEGER (2 Byte) yang berisikan informasi panjang dan lebar dari gambar pada awal memori dengan alamat pointer SPRITE. Tentu saja panjang dari gambar tersebut ialah (X2-X1+1) dan lebarnya adalah (Y2-Y1+1). Setelah itu, mulai byte ke-5 dan seterusnya berisi data dari gambar yang direkam.

```
Procedure Store_Sprite;
  Begin
    GetMem(Sprite,ImageSize(10,10,110,110));
    GetImage(10,10,110,110,Sprite^);
  End;
```

Untuk menentukan besar gambar yang hendak direkam di dalam memori, GRAPH.TPU dari TURBO PASCAL telah menyediakan function yaitu IMAGESIZE seperti yang telah dijelaskan di atas, sehingga kita tidak perlu lagi menghitungnya sendiri menggunakan rumus di atas. Yang perlu Anda ingat adalah, besar maximum dari memori yang bisa dipakai oleh GETMEM adalah satu segmen memori (64 kbytes = 65536 bytes). Untuk itu jangan sampai function IMAGESIZE menghasilkan nilai yang lebih besar dari pada 65536.

Sekarang kita telah mempunyai gambar yang telah tersimpan dalam memori. Untuk selanjutnya gambar tersebut akan kita sebut sebagai "sprite". Proses selanjutnya menampilkan gambar tersebut ke layar dengan koordinat yang berubah-ubah. Tapi sebelum itu kita akan membersihkan layar, jadi pada awal dari procedure MAIN, digunakan statement CLEARDEVICE yang fungsinya untuk menghapus layar.

Proses menggerakkan gambar dilakukan dengan looping FOR X:=1 to 500 DO. Dan bila kita lihat pada Statement PUTIMAGE di dalam looping, parameter koordinat X dan Y sama-sama diisi dengan nilai X yang berarti koordinat X dan Y dari gambar akan bergeser mulai koordinat (1,1) hingga (500,500). Di dalam looping statement PUTIMAGE dipanggil dua kali dengan parameter yang sama, tetapi pada PUTIMAGE yang kedua, parameter model peletakan gambar dipilih operasi XORPUT yang akan menghapus gambar yang telah ada di layar.

```
Procedure Main;
  Begin
    ClearDevice;
    For X:=1 to 500 do
      Begin
        PutImage(X,X,Sprite^,NormalPut);
        Delay(10);
        PutImage(X,X,Sprite^,XorPut);
     End;
 End;
```

Di antara kedua PUTIMAGE ini, diberikan DELAY(10) yang bertujuan untuk memberikan selang waktu antara penggambaran dan penghapusan gambar. Coba hilangkan DELAY ini dan jalankan program di atas. Anda akan melihat bahwa animasi terlihat berkedip-kedip. Ini dikarenakan pada saat gambar dihapus, mata kita melihat layar hitam kosong. Dan karena proses ini ada di dalam looping, maka Anda akan mendapati bahwa animasi akan berkedip.

PEMBUATAN GAME SEDERHANA DENGAN GRAPH.TPU

Pembuatan game juga termasuk bagian dari animasi. Proses pembuatannya tidak jauh berbeda dari program animasi. Hanya saja, di

sini Anda dituntut untuk membuat animasi itu menjadi aktif bukannya pasif. Maksud dari aktif adalah sebagian dari gerakkan animasi itu dikontrol secara langsung oleh pemain melalui keyboard, mouse ataupun joystick tidak seperti animasi-animasi pada program-program kita di atas yang hanya tinggal dijalankan dan untuk selanjutnya kita tinggal menonton tanpa bisa menggerakkan suatu apa pun.

Satu hal lagi mengenai perbedaan yang menyolok antara game dan animasi adalah pendeteksian dan penganalisaan status dari tiap object. Kebanyakan dari Anda tentunya tahu permainan SPACE INVADER, Anda harus berlindung dari serangan musuh dengan menempatkan pesawat Anda di bawah kubu pertahanan, disamping itu Anda juga harus menembakkan laser ke atas untuk menghancurkan musuh. Bagaimana komputer bisa tahu kalau peluru kita berhasil mengenai musuh, tembakan musuh mengenai kubu, tembakan musuh mengenai pesawat kita? Semua itu dilakukan dengan cara menganalisa dan mendeteksi koordinat dari setiap object. Tentu saja untuk animasi, hal-hal yang seperti ini tidak terlalu dibutuhkan karena animasi itu akan selalu sama. Walaupun Anda jalankan besok, minggu depan ataupun tahun depan pun jalannya tetap sama. Tapi berbeda dengan pembuatan game. Ada kemungkinan Anda meledak pada saat-saat tertentu ada kemungkinan pula Anda memenangkan permainan.

Sebaiknya Anda pelajari dahulu program di bawah ini. Inilah program game buatan kita yang pertama kali.

Listing Program

```
{ ------------------------------------------------------------ }
{ File      :  Anim4.Pas                                       }
{ Compiler  :  Turbo Pascal 7.0                                }
{ Tujuan    :  Permainan Memukul Semut                         }
{  ------------------------------------------------------------}

Uses Crt,Graph,Dos;
Const
    Up      =  1;
    Down  =  2;
    Left     =  3;
    Right   =  4;
    Hit      =   5;
Var X,Y                          : Integer;
       Gr_Mode,Sc_Mode: Integer;
       Dir                       : Byte;
       Ch                        : Char;
```

```
      Luar                    : Boolean;
      Live                    : Boolean;
      Sprite                  : Pointer;
      Semut                   : Pointer;
      X_L,Y_L                 : Integer;
      Score                   : Byte;
      A,B,Timer               : Integer;

Procedure Initialize;
  Begin
    Gr_Mode:= Vga;
    Sc_Mode:= VgaHi;
    InitGraph(Gr_Mode,Sc_Mode,'');
  End;

Procedure Draw_Sprite;
  Begin
    SetColor(15);
    Circle(60,60,50);
    SetFillStyle(2,8);
    FloodFill(60,60,15);
    Rectangle(58,110,62,150);
    Circle(300,100,5);                { Badan        }
    Circle(300,90,3);                 { Kepala       }
    SetFillStyle(1,6);
    FloodFill(300,100,15);
    SetFillStyle(1,14);
    FloodFill(300,90,15);
    Line(296,96,290,90);        { Kaki Kiri 1  }
    Line(295,98,288,98);        { Kaki kiri 2  }
    Line(296,100,290,106);      { Kaki kiri 3  }
    Line(304,96,310,90);        { Kaki kanan 1         }
    Line(305,98,312,98);        { Kaki kanan 2         }
    Line(304,100,310,106);      { Kaki kanan 3         }
  End;

Procedure Store_Sprite;
  Begin
    GetMem(Sprite,ImageSize(10,10,110,150));
    GetImage(10,10,110,150,Sprite^);
    GetMem(Semut,Imagesize(285,86,315,110));
    GetImage(285,86,315,110,Semut^);
    ClearDevice;
  End;

Procedure Place_Target;
  Begin
    Inc(Timer);
    If Timer>50 then
```

```
      Begin
        Timer:=0;
        Live:=True;
        X_L:=Round(Random(500))+70;
        Y_L:=Round(Random(340))+70;
      End;
  End;

Procedure Hit_Target;
  Begin
    If (X_L<X+100) and (X_L+30>X) then
    If (Y_L<Y+140) and (Y_L+24>Y) then
    If Live then
      Begin
        Sound(50);
        Delay(30);
        Sound(80);
        Delay(30);
        Sound(200);
        Inc(Score);
        Circle(10+Score*20,6,5);
        SetFillStyle(1,6);
        FloodFill(10+Score*20,6,15);
        Live:=False;
     End Else Sound(60);
    Delay(50);
    Nosound;
    Dir:=0;
 End;

Procedure Do_Noise;
  Begin
    If Timer Mod 20 in [0..10] then
      Sound(100) else Sound(80);
  End;

Procedure Main;
  Begin
    Luar:=False;
    Live:=True;
    X:=320;Y:=200;
    Dir:=0;
    Timer:=0;
    X_L:=100;Y_L:=100;
    Score:=0;
    Repeat
      Repeat
        Delay(5);
        PutImage(X,Y,Sprite^,XorPut);
        If Live then PutImage(X_L,Y_L,Semut^,XorPut);
```

```
        Case Dir of
           Up   :If Y>30 then Dec(Y,10);
           Down:If Y<330 then Inc(Y,10);
           Left:If X>10 then Dec(X,10);
           Right:If X<530 then Inc(X,10);
        End;
        Place_Target;
        If Live then Do_Noise;
        If Dir=Hit then Hit_Target;
        If Live then PutImage(X_L,Y_L,Semut^,NormalPut);
        PutImage(X,Y,Sprite^,NormalPut);
      Until (Keypressed) or (Score=20);
      Ch:=Readkey;
      Case Ch of
         #72:Dir:=Up;
         #80:Dir:=Down;
         #75:Dir:=Left;
         #77:Dir:=Right;
         #27:Luar:=True;
         ' ':Dir:=Hit;
      End;
  Until (Luar) or (Score=20);
  Nosound;
End;

Procedure Do_Result;
  Begin
    Writeln('Anda memang pemukul semut yang canggih !');
  End;

Begin
  Initialize;
  Draw_Sprite;
  Store_Sprite;
  Main;
  CloseGraph;
  Do_Result;
End.
```

Penjelasan Listing Program

Program di atas adalah program permainan memukul semut. Anda akan diberikan alat pemukul yang dapat Anda gerakkan dengan tombol kursor dan untuk memukul digunakan tombol Space Bar. Untuk keluar dari permainan tekan tombol Escape.

Pada awal program dijalankan Anda akan melihat pemukul semut Anda dan seekor semut. Gerakkan pemukul semut sehingga semut telah

berada di bawah pemukul lalu tekan Space Bar. Anda harus bergerak cepat karena semut tidak akan diam saja di satu tempat, melainkan akan pindah di tempat lain dengan cepat setelah beberapa detik muncul. Setelah Anda berhasil membunuhnya, akan keluar semut lain. Lingkaran coklat di atas layar menunjukkan jumlah semut yang telah Anda pukul. Permainan akan berakhir setelah Anda menekan tombol Escape atau Anda telah membunuh 20 semut.

Ini merupakan contoh dari game yang sederhana. Dengan ini Anda akan mendapatkan gambaran tentang pembuatan game. Baiklah kita mulai saja membahas program di atas.

STRUKTUR VARIABEL

Sebelumnya marilah kita pelajari dahulu beberapa variabel penting yang digunakan pada program di atas:

- X dan Y digunakan untuk menyimpan koordinat dari pemukul semut yang kita gerakkan.
- X_L dan Y_L digunakan untuk menyimpan koordinat semut.
- Live digunakan sebagai status apakah semut masih hidup atau sudah mati.
- Score digunakan untuk menampung score dari pemain.
- Dir menunjukkan tombol perintah yang kita tekan.
- Timer menunjukkan berapa lama seekor semut telah berada pada suatu tempat.
- Sprite adalah variabel pointer yang menampung gambar dari pemukul semut.
- Semut adalah variabel pointer yang menampung gambar dari semut.

STRUKTUR PROGRAM

- Inisialisasi.
- Main.
    - Inisialisasi Variabel

- Looping Pertama.
    - Looping Kedua.
        - Menghapus Gambar.
        - Memindahkan Posisi Gambar.
        - Menganalisa Situasi.
        - Meletakkan Gambar.
    - Deteksi Tombol.
- Akhir.

PENJELASAN STRUKTUR PROGRAM

Inisialisasi

Pada awal program seperti biasa dipanggil procedure INITIALIZE untuk meset mode layar ke mode Graphic VGAHI. Selanjutnya memanggil procedure DRAW_SPRITE untuk merancang gambar pemukul semut dan semut.

```
Procedure Draw_Sprite;
  Begin
    SetColor(15);
    Circle(60,60,50);
    SetFillStyle(2,8);
    FloodFill(60,60,15);
    Rectangle(58,110,62,150);
    Circle(300,100,5);                { Badan          }
    Circle(300,90,3);                 { Kepala         }
    SetFillStyle(1,6);
    FloodFill(300,100,15);
    SetFillStyle(1,14);
    FloodFill(300,90,15);
    Line(296,96,290,90);        { Kaki Kiri 1          }
    Line(295,98,288,98);        { Kaki kiri 2          }
    Line(296,100,290,106);      { Kaki kiri 3          }
    Line(304,96,310,90);        { Kaki kanan 1         }
    Line(305,98,312,98);        { Kaki kanan 2         }
    Line(304,100,310,106);      { Kaki kanan 3         }
 End;
```

Selanjutnya, seperti program-program kita terdahulu dilakukan perekaman gambar dari layar ke memori dengan menggunakan statement GETIMAGE. Untuk gambar pemukul semut dimasukkan ke dalam alamat pointer SPRITE sedangkan untuk gambar semut ke alamat pointer SEMUT.

```
Procedure Store_Sprite;
  Begin
    GetMem(Sprite,ImageSize(10,10,110,150));
    GetImage(10,10,110,150,Sprite^);
    GetMem(Semut,Imagesize(285,86,315,110));
    GetImage(285,86,315,110,Semut^);
    ClearDevice;
End;
```

MAIN

Inisialisasi Variabel

Pada bagian awal dari procedure MAIN, variabel-variabel tertentu seperti Score, X,Y,X_L,Y_L,Live diberi nilai awal. Di sini diatur koordinat awal dari pemukul semut adalah (320,200) sedangkan semut pertama akan muncul di koordinat (100, 100). Status Live dari semut adalah TRUE yang berarti hidup. Score Anda masih 0 dan Timer juga 0.

```
Procedure Main;
  Begin
    Luar:=False;
    Live:=True;
    X:=320;Y:=200;
    Dir:=0;
    Timer:=0;
    X_L:=100;Y_L:=100;
    Score:=0;
```

Menghapus Gambar

Perhatikan pada Loop yang kedua yaitu Repeat ..... Until (KeyPressed) or (Score=20). Program tidak akan keluar dari loop ini sebelum Anda mencapai score 20 atau Anda menekan sesuatu.

Pada awal loop terdapat statement DELAY yang berguna untuk memberikan selang waktu antara penggambaran dengan penghapusan gambar seperti yang telah dijelaskan di atas. Setelah itu dilakukan penggambaran penghapusan gambar yang ada di layar yaitu dengan PUTIMAGE mode XORPUT. Untuk penghapusan semut diberi testing dahulu apakah semut masih hidup, sebab apabila semut sudah mati, di layar tidak mungkin ada gambar semut.

```
Delay(5);
PutImage(X,Y,Sprite^,XorPut);
If Live then PutImage(X_L,Y_L,Semut^,XorPut);
```

Memindahkan Posisi Gambar

Berikutnya terdapat pemeriksaan nilai dari variabel DIR. Seperti yang telah kita ketahui bahwa nilai variabel ini menunjukkan tombol perintah yang kita tekan dan nilai dari perintah itu telah kita masukkan ke dalam konstanta. Up berarti 1, Down berarti 2 dan sebagainya. Sehingga kita dapat merubah nilai koordinat X dan Y sesuai dengan yang diperintahkan. Bila Dir berharga 1 atau Up maka variabel Y dikurangi, bila Dir berharga 3 atau Left maka Variabel X dikurangi. Sebelum koordinat-koordinat tersebut diubah, dilakukan pemeriksaan batas layar. Pada layar mode VGAHI, batas layar horizontal adalah 640 dan batas vertical adalah 480. Ini berarti kita tidak dapat seenaknya saja menggerakkan pemukul sampai menerobos batas layar. Untuk itu dlakukan pemeriksaan dahulu dan jika ternyata pemukul tersebut masih berada di bawah batas layar proses perubahan koordinat dilakukan.

```
Case Dir of
   Up   :If Y>30 then Dec(Y,10);
   Down:If Y<330 then Inc(Y,10);
   Left:If X>10 then Dec(X,10);
   Right:If X<530 then Inc(X,10);
End;
```

Pada PUTIMAGE, parameter X dan Y digunakan untuk menunjukkan posisi titik kiri atas dari gambar pada layar monitor.

Untuk memeriksa batas atas layar (Dir = Up) nilai Y SETELAH DIKURANGI tidak boleh kurang dari 30 karena mulai koordinat Y = 0 sampai Y = 20 dipergunakan untuk Score dari permainan.

Untuk memeriksa batas bawah dari layar (Dir = Down) nilai Y SETELAH DITAMBAH tidak boleh melebihi 360 walaupun batas koordinat Y maximum khan masih 480. Ini dikarenakan koordinat Y pada gambar itu mewakili koordinat Y dari titik kiri atas gambar. Maka bila kita ingin semua gambar dari pemukul semut tertampung dalam layar setelah penambahan koordinat Y dilakukan koordinat Y dari BAWAH gambar harus kurang dari 480. Sedangkan Koordinat BAWAH gambar adalah koordinat Y dari Atas Gambar + Lebar dari Gambar atau secara matematis.

Karena : Yb<480+P dan Yb=Ya+Lebar
Maka : Ya+Lebar <480+P
Ya<480-Lebar+P

Yb	=	Koordinat Y bawah gambar.
Ya	=	Koordinat Y atas gambar.
Lebar	=	Lebar gambar = 140.
P	=	Penambahan koordinat gambar = 10.

Untuk pemeriksaan kiri layar sama dengan pemeriksaan atas layar dimana koordinat X kiri gambar setelah dikurangi tidak boleh kurang dari 0.

Untuk pemeriksaan kanan layar sama juga dengan pemeriksaan bawah layar dimana koordinat kanan gambar tidak boleh melebihi 640.

Karena : Xb<640 dan Xb=Xa+Panjang
Maka : Xa+Panjang<640+P
Xa<640-Panjang+P

Xb	=	Koordinat X kanan gambar.
Xa	=	Koordinat X kiri gambar.
Panjang	=	Panjang gambar = 100.
P	=	Penambahan koordinat gambar = 10.

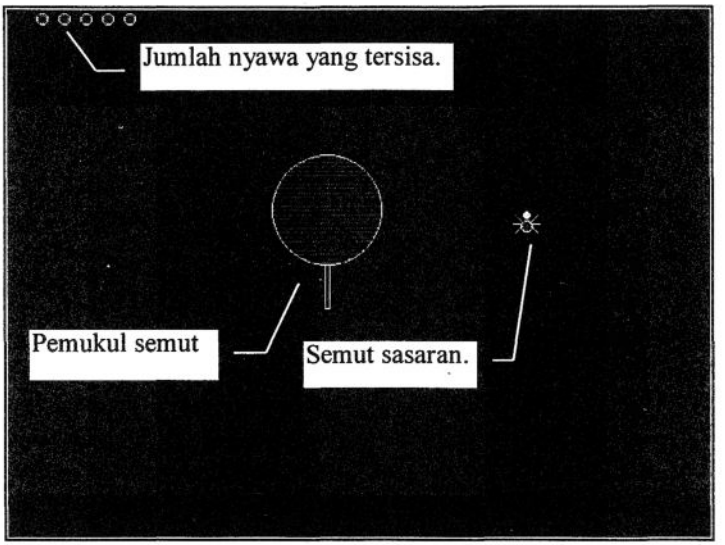
Proses selanjutnya ialah procedure PLACE_TARGET. Pada setiap kali procedure ini dipanggil, variabel TIMER akan selalu ditambah. Pada saat Timer lebih besar dari 50 dilakukan proses pemindahan letak Semut. Untuk itu Timer diset kembali ke 0, nilai variabel Live diisi dengan TRUE yang menunjukkan bahwa semut hidup. Sedangkan X_L dan Y_L yang merupakan koordinat dari semut diisi dengan bilangan random. Untuk nilai X_L akan berkisar antara 70 sampai 570 sedangkan nilai Y_L akan berkisar antara 70 sampai 410. Ini bertujuan untuk membatasi koordinat keluarnya semut agar tidak keluar dari jangkauan pemukul.

```
Procedure Place_Target;
  Begin
    Inc(Timer);
    If Timer>50 then
      Begin
        Timer:=0;
        Live:=True;
        X_L:=Round(Random(500))+70;
        Y_L:=Round(Random(340))+70;
      End;
  End;
```

Menganalisa Situasi

Fungsi dari IF LIVE THEN DO_NOISE adalah untuk memberikan suara ribut apabila semut masih hidup.

Jika variabel DIR berisikan nilai 5 atau Hit maka procedure HIT_TARGET dipanggil. Di sini dilakukan pemeriksaan apakah semut berada di bawah pemukul semut atau tidak. Adapun system pengecekannya adalah menggunakan cara mengetahui apakah dua buah persegi panjang berpotongan atau tidak. Di sini kita misalkan pemukul dan semut adalah suatu persegi panjang.



Gambar 1.5 Output Screen dari permainan memukul semut.

Dua persegi panjang berada pada situasi berpotongan bila syarat-syarat berikut dipenuhi.

$$X1 < X2 + P2 \text{ dan } X1 + P1 > X2 \text{ dan } Y1 < Y2 + L2 \text{ dan } Y1 + L1 > L2$$

Dimana variabel:

X1	=	Koordinat X kiri dari persegi panjang pertama.
Y1	=	Koordinat Y atas dari persegi panjang pertama.

X2	=	Koordinat X kiri dari persegi panjang kedua.
Y2	=	Koordinat Y atas dari persegi panjang kedua.
P1	=	Panjang dari persegi panjang pertama.
P2	=	Panjang dari persegi panjang kedua.
L1	=	Lebar dari persegi panjang pertama.
L2	=	Lebar dari persegi panjang kedua.

Pada program ini, X1=X_L, Y1=Y_L, X2=X, Y2=Y,P1=30, L1=24, P2=100, L2=140. Memang cara pemeriksaan di atas tidak bisa akurat 100 persen karena bentuk dari pemukul maupun lalat sebenarnya bukan merupakan persegi panjang. Tetapi pada cara ini masih tetap dipakai pada banyak game-game yang dijual di pasaran.

```
Procedure Hit_Target;
  Begin
    If (X_L<X+100) and (X_L+30>X) then
    If (Y_L<Y+140) and (Y_L+24>Y) then
    If Live then
      Begin
        Sound(50);
        Delay(30);
        Sound(80);
        Delay(30);
        Sound(200);
        Inc(Score);
        Circle(10+Score*20,6,5);
        SetFillStyle(1,6);
        FloodFill(10+Score*20,6,15);
        Live:=False;
        End Else Sound(60);
        Delay(50);
        Nosound;
        Dir:=0;
      End;
```

Setelah dilakukan pemeriksaan apakah semut kena pukul, dilakukan lagi pemeriksaan apakah semut masih hidup atau sudah mati. Tentu saja programmer tidak ingin kalau semut sudah mati, masih bisa dipukul dan lagi masih bisa menghasilkan score. Apabila semut masih hidup maka Live diisi dengan nilai FALSE dan Score ditambah satu. Di samping itu dilakukan juga penggambaran tambahan lingkaran pada atas layar yang menunjukkan jumlah score yang telah kita peroleh.

Meletakkan Gambar

Langkah terakhir adalah melakukan peletakan gambar dari pemukul dan semut pada layar monitor. Untuk itu dicek lagi apakah live berharga TRUE. Jika tidak tidak perlu dilakukan peletakan gambar semut.

```
If Live then PutImage(X_L,Y_L,Semut^,NormalPut);
PutImage(X,Y,Sprite^,NormalPut);
```

Deteksi Tombol

Sekarang kita tinjau REPEAT UNTIL yang paling luar. Yaitu REPEAT ..... UNTIL (LUAR) OR (SCORE = 20);

```
Ch:=Readkey;
Case Ch of
   #72:Dir:=Up;
   #80:Dir:=Down;
   #75:Dir:=Left;
   #77:Dir:=Right;
   #27:Luar:=True;
   ' ':Dir:=Hit;
End;
```

Di sini dilakukan pemeriksaan tombol keyboard. Bila tombol kursor atas ditekan maka nilai dari variabel DIR diisi dengan 1 atau Up, bila tombol kursor bawah ditekan maka nilai dari variabel DIR diisi dengan 2 atau Down dan sebagainya. Ini ditujukan untuk memberikan nilai dari tombol perintah yang Anda tekan. Untuk keluar, apabila tombol #27 atau Escape ditekan maka nilai dari variabel Luar akan menjadi TRUE sehingga pada looping terluar pemeriksaan terhadap variabel luar akan berharga TRUE yang berarti keluar dari permainan.

Akhir

Kedua procedure terakhir ialah procedure CLOSEGRAPH untuk kembali ke mode TEXT dan DO_RESULT yang berfungsi untuk menampilkan pesan bahwa Anda telah memenangkan permainan ini.

Apabila Anda masih belum begitu mengerti mengenai logika program ini, cobalah untuk melakukan TRACING (tombol F7 pada Turbo Pascal) satu-persatu untuk menganalisa hasil output dari tiap line. Dengan ini maka Bab 1 telah berakhir dan diharapkan Anda telah memiliki dasar-dasar pemrograman game sederhana.

BAB 2

DASAR PEMROGRAMAN ANIMASI DAN GAME II

PENDAHULUAN

Selamat datang di tingkat profesional. Di sini Anda akan banyak belajar mengenai pemrograman secara langsung ke memori, interrupt maupun port yang dikenal dengan DMA (Direct Memory Access). Tentunya pada saat Anda mempelajari BAB I Anda merasa bahwa dengan GRAPH.TPU Anda bisa membuat apa saja khan? Anda bisa membuat game arcade pesawat, game space invader dan sebagainya. Memang dengan GRAPH.TPU bisa dibuat program-program seperti itu tapi dalam taraf amatir. Kita tidak mungkin dapat membuat game-game yang setingkat dengan game di pasaran seperti RAPTOR, KING'S QUEST, DOUBLE DRAGON, STREET FIGHTER, WING COMMANDER apalagi DOOM yang merupakan game 3D yang tercanggih saat ini.

Untuk membuat game-game seperti itu kita perlu memperhitungkan masalah optimasi kecepatan (kecepatan jalannya animasi), akses memori (bahkan perlu mempergunakan XMS atau Expanded memory), tampilan graphic yang mendukung dan bahkan Sound Device seperti Sound Blaster, Adlib, Midi dan lain sebagainya. Hal-hal itu sudah diluar dari GRAPH.TPU. Anda ingin membuat game setaraf dengan game profesional? Ayo terus!

KELEMAHAN-KELEMAHAN GRAPH.TPU

Apabila Anda telah mencoba program-program pada BAB I, apalagi kalau Anda sudah mencoba membuat program sendiri dengan mengandalkan GRAPH.TPU, pasti Anda mengalami banyak keterbatasan. Di antaranya adalah sebagai berikut.

KETERBATASAN MODE GRAPHIC

Pada GRAPH.TPU disediakan driver untuk CGA, EGA maupun VGA. Tetapi mode graphic VGA yang dapat Anda pilih tidak dapat men-support 256 warna. Resolusi maximum yang dapat Anda pilih adalah 640 x 480 16 warna. Lalu bagaimana bila kita mau membuat game yang mempergunakan 256 warna. Jawabannya adalah tidak mungkin, selama Anda masih mengandalkan GRAPH.TPU.

BENTUK SPRITE

Pernahkah Anda mencoba untuk membuat animasi gambar bergerak dimana terdapat gambar background. Maksudnya bukan cuma gambar yang bergerak di atas background hitam polos, melainkan di atas background yang berisi gambar. Bila Anda telah mencobanya, sprite yang bergerak di layar akan terlihat seperti di dalam sebuah persegi panjang hitam. Hal ini tidak akan Anda dapati bila Anda mempergunakan background hitam polos. Kalau Anda belum pernah melihat kejadian ini, cobalah program berikut ini.

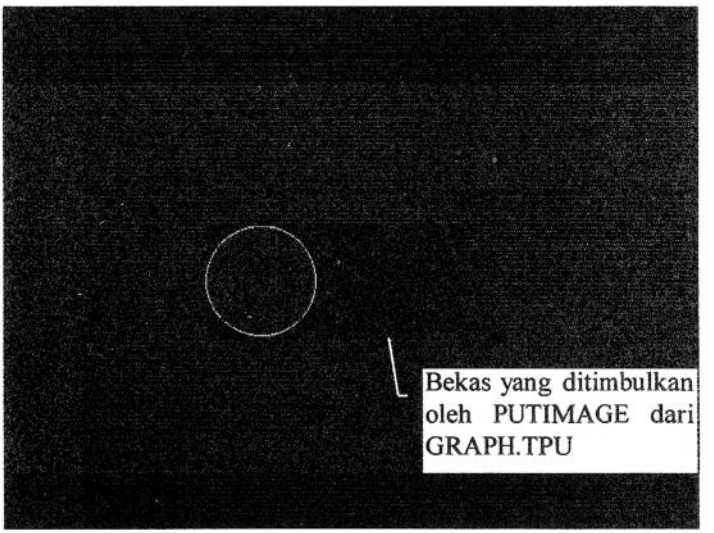
Listing Program

```
{ ------------------------------------------------------------ }
{ File        :  Bab21.Pas                                     }
{ Compiler:   Turbo Pascal 7.0                                 }
{ Tujuan    :  Menggerakkan Bola Di Layar                      }
{  ------------------------------------------------------------}
Uses Crt,Graph,Dos;

Const
    Up      = 1;
    Down = 2;
    Left    = 3;
    Right  = 4;

Var X,Y                          : Integer;
       Gr_Mode,Sc_Mode: Integer;
       Dir                          : Byte;
       Ch                           : Char;
       Luar                         : Boolean;
       Sprite                       : Pointer;

Procedure Initialize;
  Begin
    Gr_Mode:= Vga;
    Sc_Mode:= VgaHi;
    InitGraph(Gr_Mode,Sc_Mode,'');
  End;

Procedure Draw_Sprite;
  Begin
    SetColor(15);
    Circle(60,60,50);
    SetFillStyle(1,8);
    FloodFill(60,60,15);
  End;
```

```
Procedure Store_Sprite;
  Begin
    GetMem(Sprite,ImageSize(10,10,110,110));
    GetImage(10,10,110,110,Sprite^);
    ClearDevice;
  End;

Procedure DrawBackground;
  Begin
    SetFillStyle(2,5);
    FloodFill(320,200,15);
  End;

Procedure Main;
  Begin
    Luar:=False;
    X:=320;Y:=200;
    Dir:=0;
    Repeat
      Repeat
        Delay(10);
        PutImage(X,Y,Sprite^,XorPut);
        Case Dir of
           Up     :Dec(Y);
           Down:Inc(Y);
           Left   :Dec(X);
           Right :Inc(X);
        End;
        PutImage(X,Y,Sprite^,NormalPut);
     Until Keypressed;
     Ch:=Readkey;
     Case Ch of
        #72:Dir:=Up;
        #80:Dir:=Down;
        #75:Dir:=Left;
        #77:Dir:=Right;
        #27:Luar:=True;
     End;
   Until Luar;
 End;

Begin
  Initialize;
  Draw_Sprite;
  Store_Sprite;
  DrawBackground;
  Main;
  CloseGraph;
End.
```

Penjelasan Listing Program

Anda tentunya familiar dengan program ini bukan. Program ini sama dengan program ANIM4.PAS di atas cuma diberi tambahan satu procedure yaitu DRAWBACKGROUND yang berfungsi untuk mengarsir layar dengan warna pink bergaris-garis. Jalankan program ini dan cobalah untuk menggerakkan bola dengan kursor, sprite bola tersebut akan terlihat seperti berada di dalam balok hitam dan akan meninggalkan bekas di posisi lamanya.



Gambar 2.1 Output Screen dari program BAB21.PAS.

Hal ini terjadi karena statement PUTIMAGE dengan mode NORMALPUT akan menaruh sprite apa adanya pada layar monitor dan karena sprite yang kita ambil dengan GETIMAGE adalah berbentuk persegi panjang, maka bentuk itulah yang akan kita lihat di layar monitor. Kemudian untuk menghapus gambar digunakan PUTIMAGE dengan mode XORPUT. Ini akan mengakibatkan sprite di layar akan dihapus dengan warna hitam polos. Proses ini terus diulang dengan posisi sprite yang berubah-ubah sehingga meninggalkan bekas balok hitam di layar.

Lantas bagaimana pemecahannya? Cobalah untuk mengganti procedure MAIN di atas dengan procedure di bawah ini.

```
Procedure Main;
  Begin
    Luar:=False;
    X:=320;Y:=200;
    Dir:=0;
    PutImage(X,Y,Sprite^,XorPut);
    Repeat
      Repeat
      Delay(10);
      PutImage(X,Y,Sprite^,XorPut);
      Case Dir of
         Up      :Dec(Y);
         Down:Inc(Y);
         Left:Dec(X);
         Right:Inc(X);
       End;
       PutImage(X,Y,Sprite^,XorPut);
    Until Keypressed;
    Ch:=Readkey;
    Case Ch of
        #72:Dir:=Up;
       #80:Dir:=Down;
       #75:Dir:=Left;
       #77:Dir:=Right;
       #27:Luar:=True;
    End;
  Until Luar;
End;
```

Sekarang jalankan program Anda. Kali ini tidak ada bekas yang ditinggalkan bukan? Sprite yang diletakkan juga tidak berbentuk persegi panjang lagi. Tapi masalah kedua timbul, sprite yang ditampilkan di layar monitor tidak seperti yang kita harapkan. Sprite dilukis di layar dengan mode transparan. Hal ini diakibatkan oleh fungsi XORPUT yang melakukan operasi XOR setiap kali gambar akan diletakkan di layar. Dan apabila fungsi XOR itu dilakukan lagi pada sprite dengan posisi yang sama, sprite tersebut akan hilang dan berganti dengan gambar background yang lama. Kenapa kok bisa begini? Karena fungsi XOR yang diulang sama halnya dengan fungsi inverse dari matrix. Yaitu kembali ke matrix asal.

Pada akhir Bab 1 disebutkan bahwa kita tidak akan menggunakan GRAPH.TPU lagi, tapi di sini kita masih akan menyinggung sedikit mengenai masalah GRAPH.TPU karena kita akan mempelajari kerugian-kerugian memakai GRAPH.TPU ini. Tapi mulai Bab 3 kita tidak akan memakai GRAPH.TPU.

Sebenarnya masih ada cara untuk menampilkan sprite yang tidak berbekas pada layar background, tapi sprite tersebut masih tetap diletakkan berwujud persegi panjang, yaitu dengan menyimpan gambar background di bawah sprite yang akan kita letakkan sebelum proses peletakan sprite pada background dilakukan. Caranya sama dengan pada waktu kita merekam sprite ke memori, dengan GETIMAGE, dan sebelum kita meletakkan sprite pada layar background, gambar background yang telah kita rekam kita tindihkan dahulu pada layar monitor sehingga background kembali normal. Berikut adalah contoh sebuah program yang memanfaatkan sistem ini.

Listing Program

```
{------------------------------------------------------------ }
{File       :  Bab22.Pas                                     }
{Compiler   :   Turbo Pascal 7.0                             }
{Tujuan     :  Menggerakkan Bola Di Layar                    }
{------------------------------------------------------------}
Uses Crt,Graph,Dos;
Const
    Up       = 1;
    Down   = 2;
    Left      = 3;
    Right    = 4;
Var X,Y                                  : Integer;
       Gr_Mode,Sc_Mode: Integer;
       Dir                               : Byte;
       Ch                                : Char;
       Luar                              : Boolean;
       Sprite                            : Pointer;
       Background                        : Pointer;

Procedure Initialize;
  Begin
    Gr_Mode:= Vga;
    Sc_Mode:= VgaHi;
    InitGraph(Gr_Mode,Sc_Mode,'');
  End;

Procedure Draw_Sprite;
  Begin
    SetColor(15);
    Circle(60,60,50);
    SetFillStyle(1,8);
    FloodFill(60,60,15);
  End;
```

```
Procedure Store_Sprite;
  Begin
    GetMem(Sprite,ImageSize(10,10,110,110));
    GetMem(BackGround,ImageSize(10,10,110,100));
    GetImage(10,10,110,110,Sprite^);
    ClearDevice;
  End;

Procedure DrawBackground;
  Begin
    SetFillStyle(2,5);
    FloodFill(320,200,15);
  End;

Procedure Main;
  Begin
    Luar:=False;
    X:=320;Y:=200;
    Dir:=0;
    GetImage(X,Y,X+100,Y+100,Background^);
    Repeat
       Repeat
         Delay(10);
         PutImage(X,Y,Background^,NormalPut);
         Case Dir of
              Up    :Dec(Y);
              Down:Inc(Y);
              Left    :Dec(X);
              Right:Inc(X);
         End;
         GetImage(X,Y,X+100,Y+100,Background^);
         PutImage(X,Y,Sprite^,NormalPut);
      Until Keypressed;
      Ch:=Readkey;
      Case Ch of
           #72:Dir:=Up;
           #80:Dir:=Down;
           #75:Dir:=Left;
           #77:Dir:=Right;
           #27:Luar:=True;
           ' ':Dir:=0;
      End;
    Until Luar;
 End;
Begin
  Initialize;
  Draw_Sprite;
  Store_Sprite;
  DrawBackground;
```

```
  Main;
  CloseGraph;
End.
```

Penjelasan Listing Program

Model program di atas tidak banyak berbeda dengan program BAB21.PAS. Tujuannya ialah agar Anda lebih cepat mempelajari teknik baru yang diajarkan dan tidak perlu menelusuri jalannya program dari awal karena perubahan pada program ini tidak banyak.

Pada procedure STORE_SPRITE ditambah GETMEM untuk variabel pointer BACKGROUND. Di sinilah gambar background yang tepat berada di bawah sprite akan disimpan. Tentu saja besar memori yang dibutuhkan sama dengan besar memori yang dibutuhkan untuk merekam sprite ke memori.

Sekarang perhatikan procedure MAIN. Di sini terdapat sedikit perubahan pula. Sebelum REPEAT, ditambahkan statement GETIMAGE. Fungsinya adalah untuk mengambil gambar di mana sprite akan diletakkan, yaitu koordinat (X,Y) sampai (X+100, Y+100) mengingat besar sprite adalah 100 x 100.

Di dalam loop akan terjadi proses sebagai berikut:

- Letakkan gambar background yang telah direkam ke layar monitor sehingga gambar background kembali seperti sedia kala.

  Statement : PutImage(X,Y,Background^,NormalPut)

- Ubah posisi sprite dengan Case ch of .....

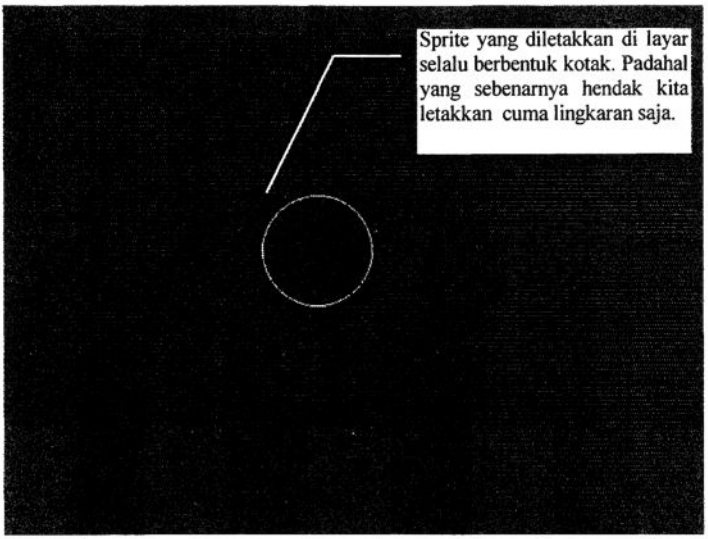
- Rekam gambar background di tempat di mana akan diletakkan sprite.

  Statement : GetImage(X,Y,X+100,Y+100,Background^)

- Letakkan gambar sprite ke posisi yang baru.

  Statement : PutImage(X,Y,Sprite^,NormalPut)

- Ulang lagi ke proses awal.



Gambar 2.2 Output Screen dari program BAB22.PAS.

Memang masalah transparansi bisa diatasi dengan cara ini, tapi seperti biasa, timbul masalah baru dimana animasi terlihat sangat berkedip. Ini disebabkan karena mata kita melihat sprite yang terhapus oleh gambar background. Sehingga kadang-kadang mata kita melihat gambar background, kadang-kadang melihat gambar sprite. Dan ternyata ini adalah upaya kita yang terakhir dan tak mungkin terpecahkan lagi. Untuk itulah kita akan beralih ke UNIT buatan sendiri dan tidak bergantung pada GRAPH.TPU.

JUMLAH SPRITE

Sebenarnya yang menjadi masalah bukan jumlah sprite yang dapat kita gerakkan di layar monitor, melainkan apa yang terjadi kalau kita menggerakkan banyak sprite pada layar monitor jika bentuk dari setiap sprite harus selalu persegi panjang. Bayangkan saja kita membuat game pesawat seperti RAPTOR. Pada satu layar ada kemungkinan Anda harus menampilkan puluhan sprite seperti ledakkan, pesawat, peluru, pecahan pesawat dan lain sebagainya. Bila semuanya berbentuk kotak hitam dengan gambar sprite di dalamnya, yang kita lihat bukanlah game pesawat lagi melainkan kotak-kotak yang berserakan di layar monitor.

Anda bisa mencobanya dengan menambahkan sejumlah sprite lagi pada program terdahulu dan coba lihat hasilnya.

KECEPATAN ANIMASI

Ini adalah masalah yang penting dalam pembuatan game. GRAPH.TPU bukanlah UNIT yang tepat untuk kita pakai bila kita hendak membuat game profesional. Coba jalankan 100 sprite sekaligus pada layar monitor dan lihat apa yang terjadi. Bila Anda memakai cara kita yang terkhir kali, yaitu dengan merekam gambar background yang hendak ditumpuk dengan sprite, Anda akan melihat kedipan-kedipan gambar di mana-mana yang menunjukkan kecepatan program menjadi sangat lambat. Dengan membuat UNIT sendiri, kita akan dapat menciptakan animasi puluhan kali lebih cepat dari GRAPH.TPU.

Sebenarnya masih banyak kelemahan-kelemahan yang terdapat pada GRAPH.TPU ini dalam pembuatan game profesional, tapi alangkah baiknya kalau kita mulai mempelajari bagaimana cara menanggulangi kelemahan-kelemahan ini dari pada terus membicarakannya.

PENGGUNAAN MODE GRAPHIC 13H

Pada halaman terdahulu telah dijelaskan bahwa GRAPH.TPU tidak mensupport mode VGA 256 warna. Sedangkan pada umumnya (kurang lebih 95 %), game-game yang ada di pasaran mengunakan mode graphic 13h dengan resolusi 320 x 200 256 warna. Mode resolusi ini banyak dipilih oleh game-game programmer seperti WING COMMANDER, COMMANCHE, DOOM, KING'S QUEST, LEMMINGS dan lain sebagainya karena di samping jumlah warna yang dapat dikeluarkan cukup banyak (256 warna), kecepatan animasi dalam mode ini cukup memadai pula. Perlu diketahui bahwa semakin besar mode resolusi yang dipakai, semakin lambat pula kecepatan animasi karena jumlah titik yang harus diakses lebih banyak. Semua game-game yang mengutamakan banyak gerakkan sprite seperti game-game yang disebutkan di atas hampir tidak mungkin untuk dibuat di resolusi tinggi, misalnya 640 x 480 256 warna ataupun 1024 x 768 256 warna. Sebenarnya sih bisa saja dengan microprocessor 80586 PENTIUM tapi hal itu akan menyebabkan pembeli game berkurang. Untuk itulah maka mulai saat ini kita akan memfokuskan program-program kita pada mode resolusi 13h.

ALOKASI PIXEL PADA MEMORI LAYAR

PENDAHULUAN

Anda tentunya masih ingat cara meletakkan titik pada layar monitor dengan bantuan PUTPIXEL dari GRAPH.TPU. Sebenarnya statement PUTPIXEL adalah merupakan procedure yang disediakan oleh Turbo Pascal. Di dalam procedure PUTPIXEL ini dilakukan kalkulasi terhadap koordinat X dan Y yang Anda berikan lewat parameter untuk mengetahui alamat memori layar yang dituju. Selanjutnya pada alamat tersebut diisikan nilai (warna) dari pixel yang Anda inginkan.

```
Program Set_Vga256;

Uses Crt,Dos;

Var Reg:Registers;

Begin
  Reg.Ax:=$10;
  Intr($10,Reg);
End;
```

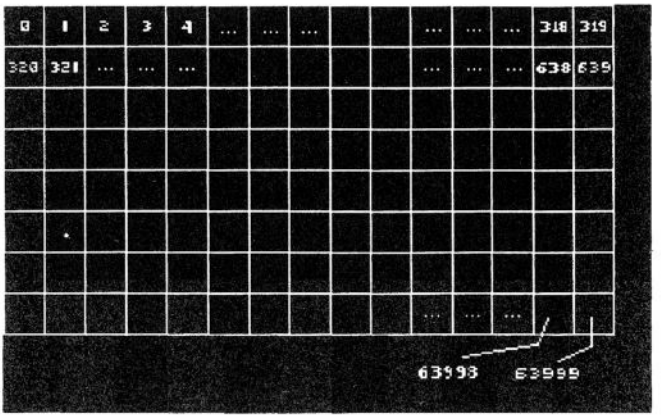
ALOKASI PIXEL SECARA LANGSUNG

Sebenarnya apa yang kita lihat pada monitor itu direkam dalam memori RAM yang sebenarnya dihasilkan oleh Card Monitor yang selalu mentransfer apa yang ada pada alamat memori tersebut pada layar monitor. Untuk selanjutnya alamat memori tersebut akan kita sebut sebagai visual page.

Sebagaimana yang telah kita ketahui, pada monitor VGA terdapat bermacam-macam mode resolusi. Setiap mode resolusi ini memiliki alamat visual page yang berbeda-beda. Untuk menggunakan mode resolusi ini kita harus menginstalnya terlebih dahulu seperti pada waktu kita menggunakan GRAPH.TPU. Untuk itu Anda dapat menggunakan interrupt DOS servis 10h. Set register AX menjadi 13h yang menunjukkan mode resolusi yang kita pilih.

Alamat visual page untuk mode resolusi 13h yaitu mulai $A000:0 sampai $A000:63999. Setiap pixel pada layar monitor disimpan ke dalam satu offset (1 offset = 1 byte). Jumlah memori yang diperlukan untuk menyimpan semua titik ini yaitu 64000 bytes karena resolusi yang

Anda pergunakan adalah 320 x 200 yang berarti diperlukan tempat penampungan pixel sejumlah 320 x 200 = 64000. Sistem alokasi pixel pada visual page ini berurutan secara horisontal dari kiri ke kanan dan dari atas ke bawah. Untuk lebih jelasnya lihatlah diagram di bawah ini.



Gambar 2.3 Sistem alokasi pixel pada monitor VGA mode $13. Setiap kotak mewakili 1 offset = 1 pixel.

Jumlah pixel horisontal adalah 320 sedangkan vertikalnya berjumlah 200 (Gambar disamping tidak menunjukkan jumlah pixel yang sebenarnya !)

Oleh karena itu maka rumus yang digunakan untuk meletakkan titik pada layar adalah sebagai berikut :

*Offset = Y*320+X*

Misalnya Anda ingin meletakkan titik pada koordinat (100, 100), Anda harus melakukan alokasi visual page ke alamat $A000:32100. Berikut adalah program untuk menampilkan titik pada mode graphic 13h.

Listing Program

```
{-------------------------------------------------------------}
{ Program   :  Screen1.Pas                                    }
{ Compiler  :  Turbo Pascal 7.0                               }
{ Tujuan    :  Menampilkan Titik Di Tengah Layar              }
{-------------------------------------------------------------}
```

```
Program Screen1;
Uses Crt,Dos;
Var Reg: Registers;
       Ch : Char;

Procedure OpenVga256;
  Begin
    Reg.Ax:=$13;                    {Interrupt Service Nomor 13 Hexa}
    Intr($10,Reg);                  {Eksekusi Interrupt              }
  End;

Procedure CloseVga256;
  Begin
    TextMode(80);                    {Kembali Ke Text Mode           }
  End;

Procedure PutPixel(X,Y: Integer; Color: Byte);
Var Address: Word;
  Begin
    Address:=(Y*320)+X;             {Terjemahkan X,Y Ke Offset      }
  Mem[$A000:address]:=Color;        {Letakkan Titik Warna = Color   }
  End;

Procedure Main;
  Begin
    PutPixel(160,100,14);           {Panggil Procedure PutPixel     }
  End;

Begin
  OpenVga256;             {Install Mode Grafik 320 x 200 256 color}
  Main;                   {Panggil Procedure Main                  }
  Ch:=Readkey;            {Tunggu Penekanan Tombol                 }
  CloseVga256;            {Kembali Ke Mode Text                    }
End.
```

Penjelasan Listing Program

Apabila program di atas Anda jalankan monitor Anda akan diset ke mode graphic 13h dan akan ditampilkan titik kuning pada tengah-tengah layar, tepatnya pada koordinat (160,100). Procedure OPENVGA berguna untuk menset layar ke mode graphic 13h dengan cara seperti yang telah dijelaskan di atas. Coba lihat pada procedure PUTPIXEL. Untuk menterjemahkan parameter X dan Y dipergunakan rumus Y*320+X. Dan setelah alamat offset diperoleh, diadakan alokasi visual page dengan MEM[$A000:Address]:=Color.

Pada mode graphic ini Anda dapat menampilkan 256 warna. Anda tidak percaya? Cobalah program berikut, Anda pasti akan terkejut.

Listing Program

```
{-------------------------------------------------------------}
{Program   : Screen2.Pas                                      }
{Compiler  : Turbo Pascal 7.0                                 }
{Tujuan    : Menampilkan Garis-Garis 256 Warna                }
{-------------------------------------------------------------}

Uses Crt,Dos;

Var Reg: Registers;
       Ch : Char;
       A,B: Integer;

Procedure OpenVga256;
  Begin
    Reg.Ax:=$13;              {Interrupt Service Nomor 13 Hexa }
    Intr($10,Reg);            {Eksekusi Interrupt              }
  End;

Procedure CloseVga256;
  Begin
    TextMode(80);             {Kembali Ke Text Mode            }
  End;

Procedure PutPixel(X,Y: Integer; Color: Byte);
Var Address: Word;
  Begin
    Address:=(Y*320)+X;       {Terjemahkan X,Y Ke Offset       }
    Mem[$A000:address]:=Color; { Letakkan Titik Warna = Color  }
  End;

Procedure Main;
  Begin
    For B:=1 to 199 do
      For A:=1 to 319 do
        PutPixel(A,B,A Mod 255); {Panggil Procedure PutPixel   }
  End;

Begin
  OpenVga256;           {Install Mode Grafik 320 x 200 256 warna}
  Main;                 {Panggil Procedure Main                  }
  Ch:=Readkey;          {Tunggu Penekanan Tombol                 }
  CloseVga256;          {Kembali Ke Mode Text                    }
End.
```

Penjelasan Listing Program

Program ini pada garis besarnya sama dengan program SCREEN1.PAS di atas, cuma diadakan sedikit modifikasi pada procedure MAIN. Di sini dilakukan dua kali looping dimana looping terdalam adalah untuk meletakkan titik secara horisontal yang diulang oleh looping terluar sehingga terbentuk balok sebesar layar dengan warna yang berbeda-beda. Apakah Anda melihat proses penggambaran balok ini per titik ataukah Anda langsung melihat balok itu muncul tiba-tiba? Kalau Anda menjawab jujur, pasti jawaban Anda yang kedua. Walaupun Anda menggunakan komputer AT 80286 pun Anda masih tidak dapat melihat proses ini. Coba Anda lakukan hal ini dengan GRAPH.TPU. Praktekan program tadi pada GRAPH.TPU dengan menggunakan PUTPIXEL dan INITGRAPH dari GRAPH.TPU. Kecepatan yang diperoleh dari alokasi memori secara langsung bila dibandingkan menggunakan unit GRAPH.TPU adalah kurang lebih 20 kali lipat.

PEMBUATAN GAME DENGAN ALOKASI MEMORI LAYAR

Coba sekarang kita membuat game sederhana dengan memanfaatkan teori-teori yang telah dijelaskan di atas yaitu dengan GETPIXEL dan PUTPIXEL yang kita buat sendiri. Cobalah untuk mempelajari program di bawah ini.

Listing Program

```
{-------------------------------------------------------------}
{Program   :  Screen3.Pas                                     }
{Compiler  :  Turbo Pascal 7.0                                }
{Tujuan    :  Game Sederhana Dengan PutPixel dan GetPixel     }
{-------------------------------------------------------------}
Uses Crt,Dos;
Const
    Left      = 1;
    Right     = 2;
    Up        = 3;
    Down      = 4;
Var Xp,Yp,A: Integer;
    Score   : Byte;
    Win     : Boolean;
```

```
    Reg     : Registers;
    Luar    : Boolean;
    Dir     : Byte;
    Hit     : Byte;
    Ctr      : Byte;
    Ch       : Char;

Procedure OpenVga256;
  Begin
    Reg.Ax:=$13;                {Interrupt Service Nomor 13 Hexa  }
    Intr($10,Reg);              {Eksekusi Interrupt               }
  End;

Procedure CloseVga256;
  Begin
    TextMode(80);               {Kembali Ke Text Mode             }
  End;

Procedure PutPixel(X,Y: Integer; Color: Byte);
Var Address: Word;
  Begin
    Address:=(Y*320)+X;         {Terjemahkan X,Y Ke Offset        }
    Mem[$A000:Address]:=Color; {Letakkan Titik Warna = Color    }
  End;
Function GetPixel(X,Y: Integer): Byte;
Var Address: Word;
  Begin
    Address:=(Y*320)+X;
    GetPixel:=Mem[$A000:Address];
  End;

Procedure Put4Block(X,Y:Integer; Color: Byte);
Var A,B:Integer;
  Begin
    For A:=X*4 to X*4+3 do
      For B:=Y*4 to Y*4+3 do
        PutPixel(A,B,Color);
  End;

Procedure Initialize;
  Begin
    Xp:=40;Yp:=25;                {Posisi Awal Pemain           }
    Score:=0;                     {Nilai = 0                    }
    Hit:=0;
    Luar:=False;
    Win:=False;
    Dir:=0;
    Ctr:=1;
    {Cetak Layar Permainan}
    For A:=1 to 20 do
```

```
      Begin

    {Cetak Penghalang}
      Put4Block(1+Round(Random(77)),1+Round(Random(45)),LightBlue);
      Put4Block(1+Round(Random(77)),1+Round(Random(45)),LightGreen);
      Put4Block(1+Round(Random(77)),1+Round(Random(45)),Brown);

    {Cetak Bonus       }
      Put4Block(1+Round(Random(77)),1+Round(Random(45)),White);
      End;

    {Cetak Batas Layar}
    For A:=0 to 79 do
      Begin
        Put4Block(A,0,LightRed);
        Put4Block(A,49,LightRed);
        Put4Block(A,47,LightRed);
      End;
    For A:=0 to 49 do
      Begin
        Put4Block(0,A,LightRed);
        Put4Block(79,A,LightRed);
      End;

    {Cetak Score & Life}
    For A:=1 to 20 do
      Begin
        Put4Block(2+A*2,48,LightGreen);
      End;
    For A:=1 to 5 do
      Begin
        Put4Block(50+A*2,48,LightBlue);
      End;
  End;

Procedure Detect_Screen;
Var Res: Byte;
  Begin
    Res:=GetPixel(Xp*4,Yp*4);
    Case Res of
      LightBlue,LightGreen,Brown:
        Begin
          Inc(Hit);
          For A:=10 downto 1 do
            Begin
              Sound(40+A*10);
              Delay(10);
              Nosound;
            End;
          Xp:=40;
```

```
          Yp:=25;
          Dir:=0;
          If Hit=5 then
            Begin
              Luar:=True;
              Win:=False;
            End;
          Put4Block(50+Hit*2,48,0);
          Delay(500);
        End;
      White:
        Begin
          Sound(200);
          Delay(40);
          Nosound;
          Inc(Score);
          If Score=20 then
            Begin
              Win:=True;
              Luar:=True;
              Delay(500);
            End;
          Put4Block(2+Score*2,48,0);
        End;
    End;
  End;

Procedure Remove_Player;
  Begin
    Put4Block(Xp,Yp,0);
  End;

Procedure Draw_Player;
  Begin
    Put4Block(Xp,Yp,Ctr);
  End;

Procedure Test_Wall;
  Begin
    If Yp<1 then Yp:=46;
    If Yp>46 then Yp:=1;
    If Xp<1 then Xp:=78;
    If Xp>78 then Xp:=1;
  End;

Procedure Move_Pos;
  Begin
    Case Dir of
      Up:Dec(Yp);
      Down:Inc(Yp);
```

```
      Left:Dec(Xp);
      Right:Inc(Xp);
    End;
  End;

Procedure Noise;
  Begin
    Sound(40+Ctr*30);
    Delay(3);
    Nosound;
  End;

Procedure Main;
  Begin
    Repeat
      Repeat
        Inc(Ctr);
        If Ctr>15 then Ctr:=1;
        Noise;
        Remove_Player;
        Move_Pos;
        Test_Wall;
        Detect_Screen;
        Draw_Player;
        Delay(40);
      Until (Keypressed) or Luar;
      If Not Luar then
        Begin
          Ch:=Readkey;
          Case ch of
            #0:Begin
                 Ch:=Readkey;
                 Case ch of
                   #72:Dir:=Up;
                   #80:Dir:=Down;
                   #75:Dir:=Left;
                   #77:Dir:=Right;
                 End;
               End;
            #27:Luar:=True;
          End;
        End;
    Until Luar;
  End;

Procedure Do_Result;
  Begin
    If Win=True then Writeln('Selamat, Anda Menang !') else
      Writeln('Anda Gagal !');
  End;
```

```
Begin
  Randomize;          {Aktifkan Bilangan Acak                        }
  OpenVga256;         {Install Mode Grafik 320 x 200 256 color      }
  Initialize;         {Preparation                                   }
  Main;               {Panggil Procedure Main                        }
  CloseVga256;        {Kembali Ke Mode Text                          }
  Do_Result;
End.
```

Penjelasan Listing Program

Pada game ini Anda akan mengendalikan kotak kecil yang berkilap yang merupakan pesawat Anda untuk mengambil 20 buah permata berwarna putih. Di sekeliling Anda terdapat banyak permata-permata yang berwarna-warni. Nyawa Anda akan dikurangi apabila Anda mengambil permata yang tidak berwarna putih. Cara mengambil permata-permata tersebut ialah dengan menabraknya. Anda diberi 5 buah nyawa untuk menelesaikan game ini.

Teknik pemrograman game di atas hampir sama dengan teknik pemrograman game memukul semut yang pernah kita buat. Ingat khan? Cuma di sini kita tidak mempergunakan perhitungan koordinat untuk mengetahui dua benda bertubrukan. Pada game ini kita mempergunakan teknik pembacaan warna titik pada visual page.

STRUKTUR VARIABEL

- Xp,Yp : Koordinat dari pesawat Anda.
- Score : Jumlah permata putih yang telah diperoleh.
- Win : Menunjukkan Anda menang atau kalah.
- Reg : Variabel Register dari DOS.
- Luar : Menunjukkan pilihan keluar.
- Dir : Arah dari pesawat Anda.
- Hit : Jumlah nyawa Anda yang hilang.
- Ctr : Variabel Counter.
- Ch : Menunjukkan tombol yang ditekan.

STRUKTUR PROGRAM

- Inisialisasi.
- Main.
  - Loop Pertama.
    - Loop Kedua

- Counter & Variasi..
- Menghapus Gambar.
- Memindahkan Posisi Gambar.
- Menganalisa Situasi.
- Meletakkan Gambar.
- Deteksi Tombol.
- Akhir.

PENJELASAN STRUKTUR PROGRAM

Inisialisasi

Pada awal program seperti biasa dilakukan initialisasi mode 13h yang dilakukan oleh procedure OPENVGA256.

```
Procedure OpenVga256;
  Begin
    Reg.Ax:=$13;        {Interrupt Service Nomor 13 Hexa      }
    Intr($10,Reg);      {Eksekusi Interrupt                   }
  End;
```

Langkah selanjutnya adalah merancang arena permainan. Pertama kali pada layar monitor dibuat 20 buath permata putih dan 60 buah permata yang berwarna biru muda, hijau muda, coklat. Posisinya diatur secara random sehingga pada setiap permainan, posisi ini selalu berubah.

```
For A:=1 to 20 do
 Begin
  {Cetak Penghalang}
  Put4Block(1+Round(Random(77)),1+Round(Random(45)),LightBlue);
  Put4Block(1+Round(Random(77)),1+Round(Random(45)),LightGreen);
  Put4Block(1+Round(Random(77)),1+Round(Random(45)),Brown);
  {Cetak Bonus}
  Put4Block(1+Round(Random(77)),1+Round(Random(45)),White);
 End;
```

Penggambaran permata digunakan procedure PUT4BLOCK yang berisi modifikasi dari PUTPIXEL yang berfungsi meletakkan titik sebesar 4x4 pada layar monitor. Sedangkan parameter yang diinputkan pada PUT4BLOCK ini bukan menunjukkan koordinat yang sesungguhnya di layar monitor melainkan koordinat berskala lompatan 4 pixel. Jadi misalkan pada parameter X dan Y kita masukkan nilai 1 dan 2 maka 4x4 titik yang ditampilkan akan berada pada koordinat (4,8) hingga (7,11)

Sistem ini dipilih karena pada game ini lompatan dari koordinat pesawat Anda adalah empat pixel. Juga besar setiap permata maupun pesawat Anda adalah 4x4 pixel. Kita tidak memakai 1x1 pixel meskipun gambar yang dihasilkan lebih halus Tentunya Anda tahu besar titik berukuran 1 pixel. Bayangkan kalau kita pakai satu pixel itu untuk pesawat kita dan juga untuk semua permata. Bagi komputer tidak menjadi masalah, karena teknik memprogramnya tinggal diubah sedikit. Tapi permainan akan menjadi menyebalkan karena tingkat kesulitan untuk mengambil ataupun menghindari permata jadi lebih tinggi karena mata kita tidak mungkin bisa menganalisa secara tepat posisi dari 1x 1 pixel sehingga pesawat kita akan sering meleset dari sasaran. Hal ini bisa disamakan dengan main TETRIS yang ukuran baloknya 1x1 pixel. Bisakah Anda menempatkan jatuhnya balok pada tempat yang Anda inginkan?

```
Procedure Put4Block(X,Y:Integer; Color: Byte);
Var A,B:Integer;
  Begin
    For A:=X*4 to X*4+3 do
      For B:=Y*4 to Y*4+3 do
        PutPixel(A,B,Color);
  End;
```

Berikutnya dilakukan proses pembuatan batas arena permainan serta indikator nyawa dan permata putih di bawah layar. Batas arena dibuat sebagai hiasan dan untuk menunjukkan batas layar permainan pada pemain. Sedangkan indikator nyawa tentu saja bertujuan untuk menunjukkan jumlah nyawa Anda, dan indikator permata putih menunjukkan jumlah permata putih yang telah Anda ambil. Semuanya ini dilakukan dengan manipulasi procedure PUT4BLOCK.

```
{Cetak Batas Layar}
    For A:=0 to 79 do
      Begin
        Put4Block(A,0,LightRed);
        Put4Block(A,49,LightRed);
        Put4Block(A,47,LightRed);
      End;
    For A:=0 to 49 do
      Begin
        Put4Block(0,A,LightRed);
        Put4Block(79,A,LightRed);
      End;
```

```
{Cetak Score & Life}
    For A:=1 to 20 do
      Begin
        Put4Block(2+A*2,48,LightGreen);
      End;
    For A:=1 to 5 do
      Begin
        Put4Block(50+A*2,48,LightBlue);
      End;
  End;
```

MAIN

Bagian utama dari program ini ialah procedure MAIN. Prinsip kerjanya adalah seperti yang biasa kita buat yaitu pendeteksian tombol keyboard yang kita tekan dan melaksanakan instruksi dari penekanan tombol tersebut.

Counter dan Variasi

Setiap satu loop dilakukan penambahan variabel Ctr (jika Ctr sudah mencapai 16, Ctr dikembalikan ke 1) yang berfungsi untuk menunjukkan warna dari pesawat Anda. Sehingga setiap saat warna pesawat Anda akan berkelip-kelip.

```
Inc(Ctr);
If Ctr>15 then Ctr:=1;
```

Procedure NOISE berfungsi untuk meramaikan suasana permainan dengan membunyikan sound dengan frekuensi Ctr*30+40 sehingga suara yang dihasilkan akan naik-turun-naik-turun.

```
Procedure Noise;
  Begin
    Sound(40+Ctr*30);
    Delay(3);
    Nosound;
  End;
```

Menghapus Gambar

Berikutnya dipanggil procedure REMOVE_PLAYER. Di sini pesawat Anda akan dihapus dari layar sebagai persiapan untuk perpindahan posisi. Penghapusan ini dilakukan dengan pemanggilan procedure PUT4BLOCK dengan parameter koordinat dari pesawat yaitu (Xp,Yp) dengan warna 0.

```
Procedure Remove_Player;
  Begin
    Put4Block(Xp,Yp,0);
  End;
```

Memindahkan Posisi Gambar

Procedure MOVE_POS berguna untuk memindahkan posisi pesawat Anda. Untuk itu, kita menggunakan CASE ... OF untuk mendeteksi perintah yang telah diinstruksikan melalui keyboard. Up, Down, Left dan Right adalah konstanta yang telah dideklarasikan pada awal program.

```
Procedure Move_Pos;
  Begin
    Case Dir of
      Up:Dec(Yp);
      Down:Inc(Yp);
      Left:Dec(Xp);
      Right:Inc(Xp);
    End;
  End;
```

Menganalisa Situasi

Tentunya kita tidak ingin pesawat kita menembus layar dan hilang dari penglihatan kita. Untuk itu dilakukan pendeteksian posisi pesawat dengan procedure TEST_WALL. Apabila posisi pesawat telah menembus atas layar maka pesawat akan dimunculkan di bawah layar. Begitu juga bila pesawat Anda menembus kiri layar, posisi pesawat Anda akan dialihkan ke kanan layar. Pendeteksian atas layar (baris pertama dari procedure TEST_WALL) koordinat pesawat dialihkan ke Yp = 46 padahal batas kanan layar adalah 319. Tentunya Anda masih ingat bahwa koordinat kita di sini mewakili empat titik sehingga sebenarnya koordinat Yp = 46 sama dengan 46*4 = 184. Koordinat Y = 184 sampai Y = 199 disediakan untuk tempat indikator nyawa dan permata putih. Hal ini dilakukan juga untuk koordinat Xp dimana batas X sebenarnya adalah 78*4 = 312.

```
Procedure Test_Wall;
  Begin
    If Yp<1 then Yp:=46;
    If Yp>46 then Yp:=1;
    If Xp<1 then Xp:=78;
    If Xp>78 then Xp:=1;
  End;
```

Proses setelah pemindahan posisi pesawat dilakukan adalah mendeteksi arena permainan apakah pada posisi pesawat yang baru itu ada object-object seperti permata. Untuk itu dipanggil procedure DETECT_SCREEN. Pemanggilan function GETPIXEL berguna untuk mendeteksi warna pixel yang ada pada posisi pesawat.

```
Procedure Detect_Screen;
Var Res: Byte;
  Begin
    Res:=GetPixel(Xp*4,Yp*4);
```

Apabila titik yang ditemukan berwarna biru muda, hijau muda atau coklat yang berarti menunjukkan keberadaan permata yang tidak boleh diambil maka dilakukan proses penambahan variabel HIT yang menunjukkan jumlah nyawa Anda yang telah hilang.

```
    Case Res of
      LightBlue,LightGreen,Brown:
        Begin
          Inc(Hit);
```

Sound efek dibunyikan untuk meramaikan suasana dan sebagai tanda bagi pemain bahwa dia telah kehilangan satu nyawa.

```
          For A:=10 downto 1 do
            Begin
              Sound(40+A*10);
              Delay(10);
              Nosound;
            End;
```

Kemudian koordinat pesawat dikembalikan ke tengah layar permainan yaitu koordinat (40, 25). Variabel DIR diisi nilai 0 yang menunjukkan tidak ada instruksi untuk bergerak, sehingga pesawat Anda tidak akan bergerak sebelum Anda menekan tombol keyboard.

```
          Xp:=40;
          Yp:=25;
          Dir:=0;
```

Apabila nyawa Anda telah hilang sebanyak lima nyawa maka permainan akan berakhir dan Anda dinyatakan kalah. Untuk itu maka

variabel LUAR diisi TRUE sehingga looping akan berakhir. Variabel WIN diisi nilai FALSE yang berfungsi untuk menyatakan Anda kalah.

```
If Hit=5 then
  Begin
    Luar:=True;
    Win:=False;
  End;
```

Gambar indikator nyawa Anda akan dikurangi 1 yaitu dengan menghapus sebuah gambar nyawa. Di sini dilakukan dengan procedure PUT4BLOCK dengan parameter 50+Hit*2 yang menunjukkan koordinat X dari gambar nyawa yang akan dihapus.

```
    Put4Block(50+Hit*2,48,0);
    Delay(500);
End;
```

Jika permata yang Anda ambil berwarna putih maka hidupkan sound efek dan tambah variabel SCORE.

```
White:
  Begin
    Sound(200);
    Delay(40);
    Nosound;
    Inc(Score);
```

Jika Variabel SCORE telah mencapai 20 artinya sudah tidak ada lagi permata putih yang tersisa, maka Anda dinyatakan menang dengan mengisi variabel WIN dengan TRUE dan permainan berakhir dengan mengisi variabel LUAR dengan TRUE.

```
If Score=20 then
  Begin
    Win:=True;
    Luar:=True;
    Delay(500);
  End;
```

Hapus gambar indikator permata putih di layar dengan procedure PUT4BLOCK dengan parameter koordinat X = 2 + Score*2 yang menunjukkan posisi gambar indikator permata putih yang akan dihapus.

```
        Put4Block(2+Score*2,48,0);
      End;
  End;
End;
```

Meletakkan Gambar

Langkah selanjutnya ialah meletakkan gambar pada layar monitor yang dilakukan dengan procedure DRAWPLAYER. Untuk itu dipanggil procedure PUT4BLOCK dengan parameter koordinat pemain dengan warna variabel CTR.

```
Procedure Draw_Player;
  Begin
    Put4Block(Xp,Yp,Ctr);
  End;
```

Mendeteksi Tombol

Apabila variabel LUAR berharga FALSE maka lakukan pemeriksaan tombol keyboard dengan READKEY.

```
If Not Luar then
      Begin
        Ch:=Readkey;
        Case ch of
          #0:Begin
               Ch:=Readkey;
               Case ch of
```

Isi nilai variabel DIR sesuai dengan tombol yang ditekan. Variabel ini berfungsi untuk menunjukkan instruksi arah dari gerakan pesawat. Untuk keluar dari permainan cukup dengan menekan tombol ESCAPE yang memiliki kode ASCII #27. Apabila Variabel CH bernilai #27 maka permainan berakhir dengan mengisikan variabel LUAR dengan TRUE.

```
                 #72:Dir:=Up;
                 #80:Dir:=Down;
                 #75:Dir:=Left;
                 #77:Dir:=Right;
               End;
             End;
          #27:Luar:=True;
        End;
      End;
```

Akhir

Sebagai penutup permainan ditampilkan pesan apakah Anda menang atau gagal. Anda menang apabila nilai variabel WIN = TRUE dan kalah bila WIN = FALSE.

```
Procedure Do_Result;
  Begin
    If Win=True then Writeln('Selamat, Anda Menang !') else
      Writeln('Anda Gagal !');
  End;
```

Sejauh ini tentunya Anda masih bisa mengikuti bukan? Kalau Anda mengalami kesulitan, pelajarilah lagi bagian STRUKTUR PROGRAM dan cobalah untuk memodifikasi sendiri program di atas.

APLIKASI PALETTE PADA MONITOR VGA

Pada monitor VGA khususnya pada mode 13h dikenal sistem palette. Tentunya Anda sudah tahu kan bahwa jumlah warna yang dapat kita pilih adalah 256 warna. 256 warna itu masing-masingnya dihasilkan dari campuran RGB (Merah, Hijau dan Biru). Misalnya warna putih terdiri dari campuran Merah, Hijau dan Biru yang sama proporsinya. Intensitas RGB ini dapat kita atur sedemikian rupa sehingga kita dapat menghasilkan variasi warna yang amat sangat banyak. Adapun tingkat intensitas warna ini adalah mulai dari 0 hingga 63 baik untuk merah, hijau maupun biru.

Jadi yang sebenarnya Anda alokasikan ke memori layar itu adalah nomor warna pada tabel PALETTE. Sistem PALETTE ini sangat berguna sekali untuk pemrograman game dan animasi. Pernahkah Anda melihat permainan komputer di mana layar dapat redup kemudian terang lagi, atau seluruh layar berubah menjadi merah secara perlahan kemudian kembali normal (kayak di DOOM waktu pemain kena tembak). Ini semua dihasilkan dari permainan warna PALETTE.

Untuk men-set warna PALETTE, DOS telah menyediakan fasilitas Interrupt 10h service AX=$1010.
Dimana Register :

Input :

Ax	=	$1010
Bl	=	Nomor Palette yang akan diubah.
Dh	=	Intensitas Merah.
Ch	=	Intensitas Hijau.
Cl	=	Intensitas Biru.

Selain itu untuk membaca warna PALETTE, disediakan pula fasilitas Interrupt 10h service Ax=$1015.
Dimana Register:

Input :

Ax	=	$1015
Bl	=	Nomor Palette yang akan dibaca.

Output :

Dh	=	Intensitas Merah.
Ch	=	Intensitas Hijau.
Cl	=	Intensitas Biru.

Sebagai latihan, lihatlah program di bawah ini.

Listing Program

```
{-------------------------------------------------------------}
{File     :  Palette1.Pas                                     }
{Compiler :  Turbo Pascal 7.0                                 }
{Tujuan     :  Mengganti Palette                              }
{-------------------------------------------------------------}
Program Palette1;
Uses Crt,Dos;
Type Pal_Cel = Record
       R,G,B: Byte;
     End;
Var A  : Integer;
    Reg: Registers;
    Ch : Char;
    Pal: Array[0..255] of Pal_Cel;

Procedure OpenVga256;
  Begin
    Reg.Ax:=$13;              {Interrupt Service Nomor 13 Hexa}
    Intr($10,Reg);            {Eksekusi Interrupt             }
  End;

Procedure CloseVga256;
  Begin
    TextMode(80);             {Kembali Ke Text Mode           }
  End;

Procedure Main;
  Begin
    For A:=0 to 191 do
      Begin
        FillChar(Mem[$A000:A*320],320,A);
      End;
  End;

Procedure SetPalette(No,R,G,B:Byte);
  Begin
    Reg.Ax:=$1010;
    Reg.Bl:=No;
    Reg.Dh:=R;
    Reg.Ch:=G;
    Reg.Cl:=B;
    Intr($10,Reg);
  End;
```

```
Procedure GetPalette(No:Byte;Var R,G,B: Byte);
  Begin
    Reg.Ax:=$1015;
    Reg.Bl:=No;
    Intr($10,Reg);
    R:=Reg.Dh;
    G:=Reg.Ch;
    B:=Reg.Cl;
  End;

Procedure Store_Palette;
  Begin
    For A:=0 to 255 do
      With Pal[A] do
        Begin
          GetPalette(A,R,G,B);
        End;
  End;

Procedure Change_Palette;
  Begin
    For A:=0 to 63 do
      Begin
        SetPalette(A,A,0,0);
        SetPalette(A+64,63-A,0,0);
        SetPalette(A+128,A,0,0);
      End;
  End;

Procedure Restore_Palette;
  Begin
    For A:=0 to 255 do
      With Pal[A] do
        Begin
          SetPalette(A,R,G,B);
        End;
  End;

Begin
  OpenVga256;           {Install Mode Grafik 320 x 200 256 color}
  Main;                 {Panggil Procedure Main                  }
  Ch:=Readkey;
  Store_Palette;
  Change_Palette;
  Ch:=Readkey;          {Tunggu Penekanan Tombol                 }
  Restore_Palette;
  Ch:=Readkey;
  CloseVga256;          {Kembali Ke Mode Text                    }
End.
```

Penjelasan Listing Program

Program ini akan melakukan penggambaran garis horisontal mulai dari atas layar Y=0 ke bawah layar Y = 191 dengan warna 0 hingga 191. Setelah itu tekan Enter, maka palette akan diubah menjadi merah gelap-terang-gelap-terang. Coba tekan Enter lagi, palette akan dikembalikan ke palette default (palette awal).

STRUKTUR VARIABEL

- A : Variabel counter.
- Reg : Variabel register DOS.
- Ch : Variabel Char untuk menerima READKEY.
- Pal : Array 0 sampai 255 dari Pal_Cel yang fungsinya untuk menyimpan intensitas RGB Palette.

PENJELASAN PROGRAM

Pada awal program dilakukan instalasi mode graphic 13h.

```
Begin
  OpenVga256;          { Install Mode Grafik 320 x 200 256 color }
```

Kemudian dipanggil procedure MAIN. Di sini dilakukan penggambaran garis horisontal pada layar monitor. Untuk itu digunakan statement FILLCHAR dari TURBO PASCAL untuk mengalokasi data sejumlah 320 byte (jumlah pixel horisontal = 320) pada tiap koordinat Y mulait Y=0 sampai Y = 191 dan warna yang dialokasikan adalah A. Di sini kita tidak memakai PUTPIXEL buatan kita sendiri yang dilooping horisontal dan vertical dengan alasan kecepatan yang diperoleh akan sangat rendah mengingat setiap saat kita memanggil procedure PUTPIXEL, dilakukan proses kalkulasi address (offset).

```
Procedure Main;
  Begin
    For A:=0 to 191 do
      Begin
        FillChar(Mem[$A000:A*320],320,A);
      End;
  End;
```

Langkah selanjutnya ialah penyimpanan default palette dengan cara membacanya dengan GETPALETTE kemudian merekamnya di array PAL. Untuk itu dipanggil procedure STORE_PALETTE.

```
Procedure Store_Palette;
  Begin
    For A:=0 to 255 do
      With Pal[A] do
        Begin
          GetPalette(A,R,G,B);
        End;
  End;
```

Setelah default palette direkam, kita mengubah intensitas palette mulai palette 0 hingga 191. Proses ini baru dilakukan setelah kita menekan sebuah tombol. Tidak semua palette (0-255) kita ubah karena kita hanya menampilkan garis dengan warna 0 hingga 191 jadi tidak perlu merubah intensitas palette di atas 191. Dipanggil procedure CHANGE_PALETTE.

```
Procedure Change_Palette;
  Begin
    For A:=0 to 63 do
      Begin
```

Intensitas Palette mulai 0 sampai 63 diisi dengan Red = A, Green = 0, Blue = 0 sehingga warna Palette 0 sampai 63 menjadi merah gelap hingga merah terang.

```
SetPalette(A,A,0,0);
```

Intensitas Palette mulai 64 sampai 127 diisi dengan Red = 63-A, Green = 0, Blue = 0 sehingga warna Palette 64 sampai 127 menjadi merah terang hingga merah gelap.

```
SetPalette(A+64,63-A,0,0);
```

Intensitas Palette mulai 128 sampai 191 diisi dengan Red = A, Green = 0, Blue = 0 sehingga warna Palette 128 sampai 191 menjadi merah gelap hingga merah terang.

```
        SetPalette(A+128,A,0,0);
      End;
  End;
```

Langkah terakhir setelah dilakukan penekanan tombol sekali lagi, palette dikembalikan ke default palette dengan cara memanggil procedure SETPALETTE dengan parameter apa yang telah direkam pada array PAL

```
Procedure Restore_Palette;
  Begin
    For A:=0 to 255 do
      With Pal[A] do
        Begin
          SetPalette(A,R,G,B);
        End;
  End;
```

Seperti biasa untuk penutup kita adakan proses penutupan mode graphic 13h dengan memanggil procedure CLOSEVGA256.

Selain cara interrupt, pengaturan intensitas palette dapat dilakukan pula dengan pengaksesan port secara langsung . Untuk itu VGA card menyediakan port $3c8 dan port $3c9 untuk pengaturan palette. Alamat ini berlaku untuk jenis-jenis card VGA pada umumnya. Port $3c8 digunakan untuk pengiriman nomor palette sedangkan Port $3c9 untuk pengiriman dan penerimaan intensitas warna merah, hijau dan biru secara bergantian.

Dengan sistem Port ini kecepatan yang dihasilkan kira-kira sampai 30 kali lebih cepat dari sistem interrupt. Coba gantilah procedure SETPALETTE dan GETPALETTE di atas dengan procedure di bawah ini.

```
Procedure SetPalette(No,R,G,B:Byte);
  Begin
    Port[$3c8]:=No;
    Port[$3c9]:=R;
    Port[$3c9]:=G;
    Port[$3c9]:=B;
  End;

Procedure GetPalette(No:Byte;Var R,G,B: Byte);
  Begin
    Port[$3c8]:=No;
    R:=Port[$3c9];
    G:=Port[$3c9];
    B:=Port[$3c9];
  End;
```

Untuk saat ini kita cukup mengenal saja apa yang dimaksud dengan palette. Selanjutnya pada pembuatan unit VGACAD.TPU kita akan lihat kehandalan teknik palette ini dengan lebih rinci.

BAB 3

PEMBUATAN UNIT VGACAD.TPU

PENGENALAN UNIT VGACAD.TPU

Setelah Anda berpengalaman menggunakan GRAPH.TPU tentunya puas dengan fasilitas pembuatan bangun geometri yang disediakan. Seperti pembuatan garis (LINE), lingkaran (CIRCLE), persegi panjang (RECTANGLE) bahkan fasilitas pembuatan graphic statistik (tidak dibahas pada buku ini) dan masih banyak fasilitas yang lainnya. Sekarang kita telah meninggalkan GRAPH.TPU, kita harus mulai membuat fasilitas-fasilitas tersebut dari dasar pada mode graphic 13h.

Pembuatan fasilitas-fasilitas tersebut tidaklah terlalu sulit. Kita perlu menerapkan ilmu matematika kita dalam programming. Nantinya semua fasilitas ini akan kita wujudkan dalam procedure dan function yang akan kita gabungkan menjadi satu unit yaitu VGACAD.TPU. Unit inilah yang nantinya akan memegang peranan pada pembuatan proyek-proyek kita selanjutnya.

Di bawah ini adalah listing program unit VGACAD.

Listing Program

```
{---------------------------------------}
{ Unit Animasi Graphic                  }
{ Created By Agustinus Nalwan           }
{ ------------------------------------- }

Unit VGACAD;
INTERFACE
Uses Crt,Dos;
Const Screen    : Pointer = Ptr ($A000,0);
Type Palette_Cell = Record
       R , G , B: Byte;
     End;
     Palette_Type = Array[0..255] of Palette_Cell;
     Palette_Addr = ^Palette_Type;
     Scroll_Page  = Array[1..600] of Pointer;
Var
     A,B          :  Integer;
     Reg          :  Registers;
     File_P       :  File;
     UserPalette:  Palette_Type;
     DefaultPalette:  Palette_Type;
     File_Pal     :  File;
     User_Pal     :  Palette_Addr;
```

```
    Default_Pal:  Palette_Addr;
    Active_Pal :  Palette_Addr;
    Page_Scroll:  Scroll_Page;
    Page_Width :  Integer;

{ Procedure Inisialisasi Graphic }

Procedure OpenVga256;
Procedure CloseVga256;

{ Procedure Manipulasi Layar }

Procedure FillWord      (Var PicBuff; Size:Word; Code:Byte);
Procedure MoveW         (Var Source; Var Dest; Size:Word);
Procedure Define_Page   (Var PicBuff: Pointer);
Procedure UnDefine_Page (Var PicBuff: Pointer);
Procedure Fill_Page     (Var PicBuff; Col: Byte);
Procedure MoveTo        (Var Source, Destination);

{ Procedure Konstruksi Bangun Geometri }

Procedure PutPixel (Var PicBuff; X,Y: Integer; Color: Byte);
Function  GetPixel (Var PicBuff; X,Y: Integer): Byte;
Procedure Line     (Var PicBuff; X1,Y1,X2,Y2:integer; Col: Byte);
Procedure Rectangle (Var PicBuff; X1,Y1,X2,Y2:integer; Col: Byte);
Procedure Block     (Var PicBuff; X1,Y1,X2,Y2: Integer; Col: Byte);
Procedure Circle    (Var PicBuff; X,Y: Integer; R: Integer; Col: Byte);
Procedure Ellipse   (Var PicBuff; X,Y: Integer; Rx,Ry: Integer; Col: Byte);

{ Procedure Manipulasi Sprite }

Procedure PutImage   (Var PicBuff;X,Y: Integer; Var Buffer);
Function  ImageSize  (X1,Y1,X2,Y2: Integer): Word;
Procedure GetImage (Var PicBuff; X1,Y1,X2,Y2: Integer; Var Buffer);
Procedure Load_Sprite  (File_Name:String; Var Ptr:Pointer;Var Size:Word);
Procedure      Save_Sprite                    (File_Name:String;
Ptr:Pointer;Size:Word);
Procedure Load_Picture  (File_Name:String; Var Buffer);
Procedure Save_Picture  (File_Name:String; Var Buffer);

{ Procedure Struktur Palette }

Procedure SetVisualPalette (No,R,G,B:Byte);
Procedure GetVisualPalette (No:Byte; Var R,G,B:Byte);
Procedure   SetPalette                    (Arr_Pal:Palette_Addr;
No,Rr,Gg,Bb:Byte);
Procedure GetPalette      (Arr_Pal:Palette_Addr; No:Byte; Var Rr,Gg,Bb:Byte);
Procedure Load_Palette     (Pal_Buff:Pointer; File_Name:String);
Procedure Save_Palette     (Pal_Buff:Pointer; File_Name:String);
Procedure Refresh_Palette;
```

```
Procedure Active_Palette    (Ac_Pal:Palette_Addr);
Procedure Restore_Palette;
Procedure Install_Palette   (Name:String);

{ Procedure Manipulasi Palette }

Procedure Set_All_Palette   (Pal_Buff:Palette_Addr; R,G,B: Byte);
Procedure Active_To_Palette(Pal_Dest:Palette_Addr; Del:Byte);
Function    Metamorph_Palette(Pal_Buff,  Pal_Dest:  Palette_Addr;
Rate:Byte): Boolean;

{ Procedure Scroll Layar }

Procedure Load_Scroll_Page (Name:String);
Procedure Free_Scroll_Page;
Procedure Put_Scroll_Page   (Var PicBuff; Y:Integer);

Implementation

Procedure  PutImage  (Var  PicBuff;X,Y:  Integer;  Var  Buffer);
ASSEMBLER;
  Asm
    Push ds             {Push Stack                          }
    Les di,PicBuff      {Arahkan ES:DI ke PICBUFF            }
    Lds si,Buffer       {Arahkan DS:SI ke BUFFER             }
    Mov bx,[ds:si]      {Baca Panjang                        }
    Inc si              {Offset + 1                          }
    Inc si              {Offset + 1                          }
    Mov cx,[ds:si]      {Baca Lebar                          }
    Inc si              {Offset + 1                          }
    Inc si              {Offset + 1                          }
    Mov dx,Y            {Masukkan Y ke Dx                    }
    Mov ax,dx           {Masukkan Dx ke Ax                   }
    Shl dx,1
    Shl dx,1            {Dx = DX * 4                         }
    Add dx,ax           {Dx = Dx + Ax                        }
    Shl dx,1
    Shl dx,1
    Shl dx,1
    Shl dx,1
    Shl dx,1
    Shl dx,1            {Dx = Dx * 32                        }
    Add dx,X            {Dx = Dx + X                         }
    Add di,dx           {Start Offset peletakan titik        }
    Cld                 {Counter Cx Mundur                   }
    @1:
      Mov dx,cx         {Dx = Sisa Lebar Gambar              }
      Mov cx,bx         {Cx = Sisa Panjang Gambar            }
      Push di           {Push Stack                          }
      @2:
```

```
        Lodsb           {Baca titik dari BUFFER                    }
        Or al,al        {Apakah titik = 0                          }
        Jz @3           {Kalau Y tidak perlu diletakkan            }
        Stosb           {Letakkan titik                            }
        Dec di          {Kurangi offset pembacaan titik            }
                        {karena di bawah akan ditambah lagi        }
        @3:
          Inc di        {Tambah offset pembacaan titik             }
      Loop @2           {CX-1, Ulang sampai sisa panjang gambar CX = 0 }
      Pop di            {Pop Stack                                 }
      Add di,320        {Register Di ke koordinat Y berikutnya     }
      Mov cx,dx         {Isi CX dengan sisa lebar gambar           }
      Loop @1           {CX-1, Ulang hingga sisa lebar gambar  = 0 }
      Pop ds            {Pop Stack                                 }
  End;

Function  ImageSize (X1,Y1,X2,Y2: Integer): Word;
  Begin
    ImageSize:= (X2-X1 + 1) * (Y2-Y1 + 1) + 4;
  End;

Procedure GetImage (Var PicBuff; X1,Y1,X2,Y2: Integer; Var Buffer); ASSEMBLER;
  Asm
    Push ds             {Push Stack                                }
    Lds si,PicBuff      {Arahkan DS:SI ke PICBUFF                  }
    Les di,Buffer       {Arahkan ES:DI ke BUFFER                   }
    Mov dx,Y1           {Dx = Y awal                               }
    Mov ax,dx           {Ax = Dx                                   }
    Shl dx,1
    Shl dx,1            {Dx = Dx * 4                               }
    Add dx,ax           {Dx = Dx + Ax                              }
    Shl dx,1
    Shl dx,1
    Shl dx,1
    Shl dx,1
    Shl dx,1
    Shl dx,1            {Dx = Dx * 32                              }
    Add dx,X1           {Dx = Dx + X awal                          }
    Add si,dx           {Offset = Offset + Dx                      }
    Mov bx,X2           {Bx = X akhir                              }
    Sub bx,X1           {Bx = Bx-X awal                            }
    Inc bx              {Bx = Bx + 1 = Panjang Gambar              }
    Mov [es:di],bx      {Tulis Panjang Gambar                      }
    Inc di              {Offset = Offset + 1                       }
    Inc di              {Offset = Offset + 1                       }
    Mov cx,Y2           {Cx = Y akhir                              }
    Sub cx,Y1           {Cx = Cx-Y awal                            }
    Inc cx              {Cx = Cx + 1 = Lebar Gambar                }
    Mov [es:di],cx      {Tulis Lebar Gambar                        }
    Inc di              {Offset = Offset + 1                       }
```

```
    Inc di              {Offset = Offset + 1                    }
    Cld                 {Counter Cx Mundur                      }
    @1:
      Mov dx,cx         {Dx = Sisa lebar gambar                 }
      Mov cx,bx         {Cx = Panjang gambar                    }
      Mov ax,si         {Ax = Si                                }
      Rep Movsb         {Baca titik sejumlah Cx                 }
      Mov si,ax         {Kembalikan Si ke Si awal               }
      Add si,320        {Si ke koordinat Y berikutnya           }
      Mov cx,dx         {Cx = Dx                                }
      Loop @1           {Cx-1, Ulang hingga Cx = 0              }
    Pop ds              {Pop Stack                              }
  End;

Procedure OpenVga256;
  Begin
    Reg.Ax:=$13;
    Intr($10,Reg);
  End;

Procedure MoveW(Var Source; Var Dest; Size:Word); ASSEMBLER;
  Asm
    Push ds             {Push Stack                             }
    Les di,Dest         {Arahkan ES:DI ke Dest                  }
    Lds si,Source       {Arahkan DS:SI ke Source                }
    Mov Cx,Size         {Counter Cx sebesar Size                }
    Rep MovsW           {Pindahkan DS:SI ke ES:DI               }
    Pop ds              {Pop Stack                              }
  End;

Procedure FillWord(Var PicBuff; Size:Word; Code:Byte); ASSEMBLER;
  Asm
    Push ds             {Push Stack                             }
    Les di,PicBuff      {Arahkan ES:DI ke PicBuff               }
    Mov Cx,Size         {Counter sebesar Size                   }
    Mov al,Code         {Al = Warna                             }
    Mov ah,Code         {Ah = Warna                             }
    Rep StosW           {Isi ES:DI dengan Warna                 }
    Pop ds              {Pop Stack                              }
  End;

Procedure PutPixel (Var PicBuff; X,Y: Integer; Color: Byte);
ASSEMBLER;
  Asm
    Push ds             {Push Stack                             }
    Les di,PicBuff      {Arahkan ES:DI ke PICBUFF               }
    Mov dx,Y            {Dx = Koordinat Y                       }
    Mov ax,dx           {Ax = Dx                                }
    Shl dx,1
    Shl dx,1            {Dx = Dx * 4                            }
```

```
    Add dx,ax          {Dx = Dx + Ax                          }
    Shl dx,1
    Shl dx,1
    Shl dx,1
    Shl dx,1
    Shl dx,1
    Shl dx,1           {Dx = Dx * 32                          }
    Add dx,X           {Dx = Dx + Koordinat X                 }
    Add di,dx          {Offset = Offset  + Dx                 }
    Mov al,Color       {Al = Warna titik                      }
    Stosb              {Alokasi memori layar                  }
    Pop ds             {Pop Stack                             }
  End;

Function  GetPixel (Var PicBuff; X,Y: Integer): Byte; ASSEMBLER;
  Asm
    Push ds            {Push Stack                            }
    Les di,PicBuff     {Arahkan ES:DI ke PICBUFF              }
    Mov dx,Y           {Dx = Koordinat Y                      }
    Mov ax,dx          {Ax = Dx                               }
    Shl dx,1
    Shl dx,1           {Dx = Dx * 4                           }
    Add dx,ax          {Dx = Dx + Ax                          }
    Shl dx,1
    Shl dx,1
    Shl dx,1
    Shl dx,1
    Shl dx,1
    Shl dx,1           {Dx = Dx * 32                          }
    Add dx,X           {Dx = Dx + Koordinat X                 }
    Add di,dx          {Offset = Offset  + Dx                 }
    Mov al,[es:di]     {Pindahkan warna titik ke function     }
    Pop ds             {Pop Stack                             }
  End;

Procedure Line(Var PicBuff; X1,Y1,X2,Y2:integer; Col: Byte);
Var Dx,Dy,Loop,Ctr          : Integer;
    XPos,YPos               : Real;
    Rx,Ry                   : Real;
  Begin
    Dx:=X2-X1;
    Dy:=Y2-Y1;
    XPos:=X1;YPos:=Y1;
    If Dx+Dy<>0 then
      If Abs(Dx)>Abs(Dy) then
        Begin
          Rx:=1;
          Ry:=Abs(Dy/Dx);
          Loop:=Abs(Dx);
        End Else
```

```
        Begin
          Ry:=1;
          Rx:=Abs(Dx/Dy);
          Loop:=Abs(Dy);
        End;
    If Dx<0 then Rx:=-Rx;
    If Dy<0 then Ry:=-Ry;
    PutPixel(PicBuff,Round(XPos),Round(YPos),Col);
    For Ctr:=1 to Loop do
      Begin
        XPos:=Xpos+Rx;
        YPos:=YPos+Ry;
        PutPixel(PicBuff,Round(XPos),Round(YPos),Col);
      End;
End;

Procedure Rectangle(Var PicBuff; X1,Y1,X2,Y2: integer; Col: Byte);
  Begin
    Line(PicBuff,X1,Y1,X1,Y2,Col);
    Line(PicBuff,X1,Y2,X2,Y2,Col);
    Line(PicBuff,X2,Y2,X2,Y1,Col);
    Line(PicBuff,X2,Y1,X1,Y1,Col);
  End;

Procedure Block(Var PicBuff; X1,Y1,X2,Y2: Integer; Col: Byte); ASSEMBLER;
  Asm
    Push ds             {Push Stack                          }
    Les di,PicBuff      {Arahkan DS:SI ke PICBUFF            }
    Mov dx,Y1           {Dx = Y awal                         }
    Mov ax,dx           {Ax = Dx                             }
    Shl dx,1
    Shl dx,1            {Dx = Dx * 4                         }
    Add dx,ax           {Dx = Dx + Ax                        }
    Shl dx,1
    Shl dx,1
    Shl dx,1
    Shl dx,1
    Shl dx,1
    Shl dx,1            {Dx = Dx * 32                        }
    Add dx,X1           {Dx = Dx + X awal                    }
    Add di,dx           {Offset = Offset + Dx                }
    Mov bx,X2           {Bx = X akhir                        }
    Sub bx,X1           {Bx = Bx-X awal                      }
    Inc bx              {Bx = Bx + 1 = Panjang Blok          }
    Mov cx,Y2           {Cx = Y akhir                        }
    Sub cx,Y1           {Cx = Cx-Y awal                      }
    Inc cx              {Cx = Cx + 1 = Lebar Blok            }
    Mov Al,Col          {Al = Warna                          }
    Cld                 {Counter Cx Mundur                   }
    @1:
```

```
      Mov dx,cx          {Dx = Sisa lebar Blok                }
      Mov cx,bx          {Cx = Panjang Blok                   }
      Mov si,di          {Ax = Si                             }
      Rep Stosb          {Baca titik sejumlah Cx              }
      Mov di,si          {Kembalikan Si ke Si awal            }
      Add di,320         {Si ke koordinat Y berikutnya        }
      Mov cx,dx          {Cx = Dx                             }
      Loop @1            {Cx-1, Ulang hingga Cx = 0           }
    Pop ds               {Pop Stack                           }
  End;

Procedure Circle (Var PicBuff; X,Y: Integer; R: Integer; Col: Byte);
Var Teta: Integer;
  Begin
    For Teta:=0 to 1440 do

PutPixel(PicBuff,X+Round(R*Cos(Teta*pi/720)),Y+Round(R*Sin(Teta*pi/720)),Col)
  End;

Procedure Ellipse (Var PicBuff; X,Y: Integer; Rx,Ry: Integer; Col: Byte);
Var Teta: Integer;
  Begin
    For Teta:=0 to 1440 do

PutPixel(PicBuff,X+Round(Rx*Cos(Teta*pi/720)),Y+Round(Ry*Sin(Teta*pi/720)),Col)
  End;

Procedure Load_Sprite(File_Name:String;Var Ptr:Pointer;Var Size:Word);
  Begin
    Assign(File_P,File_Name);
    Reset(File_P,1);
    GetMem(Ptr,FileSize(File_P));
    BlockRead(File_P,Ptr^,FileSize(File_P));
    Size:=FileSize(File_P);
    Close(File_P);
  End;

Procedure Save_Sprite(File_Name:String;Ptr:Pointer;Size:Word);
  Begin
    Assign(File_P,File_Name);
    Rewrite(File_P,1);
    BlockWrite(File_P,Ptr^,Size);
    Close(File_P);
  End;

Procedure Define_Page(Var PicBuff: Pointer);
  Begin
    GetMem(PicBuff,64000);
  End;
```

```
Procedure UnDefine_Page(Var PicBuff: Pointer);
  Begin
    FreeMem(PicBuff,64000);
  End;

Procedure Fill_Page (Var PicBuff; Col: Byte);
  Begin
    FillWord(PicBuff,32000,Col);
  End;

Procedure MoveTo (Var Source, Destination);
  Begin
    MoveW(Source,Destination,32000);
  End;

Procedure Load_Picture (File_Name:String;Var Buffer);
  Begin
    Assign(File_P,File_Name);
    Reset(File_P,1);
    BlockRead(File_P,Buffer,64000);
    Close(File_P);
  End;

Procedure Save_Picture (File_Name:String;Var Buffer);
  Begin
    Assign(File_P,File_Name);
    Rewrite(File_P,1);
    BlockWrite(File_P,Buffer,64000);
    Close(File_P);
  End;

Procedure SetVisualPalette (No,R,G,B:Byte);
  Begin
    Port[$3c8]:=No;        {Port nomor Palette      }
    Port[$3c9]:=R;         {Isi Port Merah          }
    Port[$3c9]:=G;         {Isi Port Hijau          }
    Port[$3c9]:=B;         {Isi Port Biru           }
  End;

Procedure GetVisualPalette (No:Byte; Var R,G,B:Byte);
  Begin
    Port[$3c8]:=No;        {Port nomor Palette      }
    R:=Port[$3c9];         {Baca Port Merah         }
    G:=Port[$3c9];         {Baca Port Hijau         }
    B:=Port[$3c9];         {Baca Port Biru          }
  End;

Procedure SetPalette    (Arr_Pal:Palette_Addr; No,Rr,Gg,Bb:Byte);
  Begin
    With Arr_Pal^[No] do
```

```
      Begin
        R:=Rr;
        G:=Gg;
        B:=Bb;
      End;
  End;

Procedure GetPalette (Arr_Pal:Palette_Addr; No:Byte; Var Rr,Gg,Bb:Byte);
  Begin
    With Arr_Pal^[No] do
      Begin
        Rr:=R;
        Gg:=G;
        Bb:=B;
      End;
  End;

Procedure Load_Palette(Pal_Buff:Pointer; File_Name:String);
Var X     : Byte;
    R,G,B : Char;
    RR,GG,BB: Byte;
  Begin
    Assign(File_Pal,File_Name);
    Reset(File_Pal,1);
    BlockRead(File_Pal,Pal_Buff^,768);
    Close(File_Pal);
  End;

Procedure Save_Palette (Pal_Buff:Pointer; File_Name:String);
Var X     : Byte;
    R,G,B : Byte;
  Begin
    Assign(File_Pal,File_Name);
    Rewrite(File_Pal,1);
    BlockWrite(File_Pal,Pal_Buff^,768);
    Close(File_Pal);
  End;

Procedure Refresh_Palette;
  Begin
    For A:=0 to 255 do
      With Active_Pal^[A] do
        Begin
          SetVisualPalette(A,R,G,B);
        End;
  End;

Procedure Active_Palette(Ac_Pal:Palette_Addr);
  Begin
    MoveW(Ac_Pal^,Active_Pal^,384);
```

```
    Refresh_Palette;
  End;

Procedure Restore_Palette;
  Begin
    MoveW(Default_Pal^,Active_Pal^,384);
  End;

Procedure Install_Palette(Name:String);
  Begin
    GetMem(Default_Pal,768);
    GetMem(User_Pal,768);
    GetMem(Active_Pal,768);
    Load_Palette(Default_Pal,Name);
    Active_Palette(Default_Pal);
  End;

Procedure Set_All_Palette    (Pal_Buff:Palette_Addr; R,G,B: Byte);
 Begin
    For A:=0 to 255 do
      SetPalette(Pal_Buff,A,R,G,B);
  End;

Procedure Active_To_Palette(Pal_Dest:Palette_Addr; Del:Byte);
Var Done   : Boolean;
    A      : Integer;
  Begin
    Repeat
      Done:=True;
      Delay(Del);
      For A:=0 to 255 do
        With Pal_Dest^[A] do
          Begin
            If Active_Pal^[A].R<R then Inc(Active_Pal^[A].R);
            If Active_Pal^[A].R>R then Dec(Active_Pal^[A].R);
            If Active_Pal^[A].G<G then Inc(Active_Pal^[A].G);
            If Active_Pal^[A].G>G then Dec(Active_Pal^[A].G);
            If Active_Pal^[A].B<B then Inc(Active_Pal^[A].B);
            If Active_Pal^[A].B>B then Dec(Active_Pal^[A].B);
            If Active_Pal^[A].R<>R then Done:=False;
            If Active_Pal^[A].G<>G then Done:=False;
            If Active_Pal^[A].B<>B then Done:=False;
          End;
      Refresh_Palette;
    Until Done;
  End;

Function   Metamorph_Palette(Pal_Buff,   Pal_Dest:   Palette_Addr;
Rate:Byte): Boolean;
```

```
  Begin
    Metamorph_Palette:=True;
      For A:=0 to 255 do
        With Pal_Dest^[A] do
          Begin
            If Pal_Buff^[A].R<R then
              Begin
                   If Pal_Buff^[A].R+Rate<R then Inc(Pal_Buff^[A].R,Rate)
                  Else Pal_Buff^[A].R:=R;
              End;
            If Pal_Buff^[A].R>R then
              Begin
                   If Pal_Buff^[A].R-Rate>R then Dec(Pal_Buff^[A].R,Rate)
                  Else Pal_Buff^[A].R:=R;
              End;
            If Pal_Buff^[A].G<G then
              Begin
                   If Pal_Buff^[A].G+Rate<G then Inc(Pal_Buff^[A].G,Rate)
                  Else Pal_Buff^[A].G:=G;
              End;
            If Pal_Buff^[A].G>G then
              Begin
                   If Pal_Buff^[A].G-Rate>G then Dec(Pal_Buff^[A].G,Rate)
                  Else Pal_Buff^[A].G:=G;
              End;
            If Pal_Buff^[A].B<B then
              Begin
                   If Pal_Buff^[A].B+Rate<B then Inc(Pal_Buff^[A].B,Rate)
                  Else Pal_Buff^[A].B:=B;
              End;
            If Pal_Buff^[A].B>B then
              Begin
                   If Pal_Buff^[A].B-Rate>B then Dec(Pal_Buff^[A].B,Rate)
                  Else Pal_Buff^[A].B:=B;
              End;
            If Pal_Buff^[A].R<>R then Metamorph_Palette:=False;
            If Pal_Buff^[A].G<>G then Metamorph_Palette:=False;
            If Pal_Buff^[A].B<>B then Metamorph_Palette:=False;
          End;
  End;

Procedure Load_Scroll_Page (Name:String);
Var Size: Longint;
    Ctr: Longint;
  Begin
    Ctr:=1;
    For A:=1 to 600 do
      GetMem(Page_Scroll[A],320);
    Assign(File_P,Name);
    Reset(File_P,1);
```

```
    Size:=FileSize(File_P);
    Page_Width:=Size div 320;
    While Size>64000 do
      Begin
        BlockRead(File_P,Page_Scroll[Ctr]^,64000);
        Size:=Size-64000;
        Inc(Ctr,200);
      End;
    BlockRead(File_P,Page_Scroll[Ctr]^,Size);
    Close(File_P);
  End;

Procedure Free_Scroll_Page;
Var Size: Word;
  Begin
    For A:=1 to 600 do
      FreeMem(Page_Scroll[A],320);
  End;

Procedure Put_Part (Var Page_Scroll; Var PicBuff; Y,Width: Integer); ASSEMBLER;
  Asm
    Push ds                 {Push Stack                          }
    Les di,PicBuff          {Arahkan ES:DI ke PICBUFF            }
    Lds si,Page_Scroll      {Arahkan DS:SI ke Page_Scroll        }
    Mov dx,Y                {Dx = Y awal                         }
    Mov ax,dx               {Ax = Dx                             }
    Shl dx,1
    Shl dx,1                {Dx = Dx * 4                         }
    Add dx,ax               {Dx = Dx + Ax                        }
    Shl dx,1
    Shl dx,1
    Shl dx,1
    Shl dx,1
    Shl dx,1
    Shl dx,1                {Dx = Dx * 32                        }
    Add di,dx               {Offset = Offset + Dx                }
    Mov dx,Width            {Dx = Lebar sisa layar               }
    Mov ax,dx               {Ax = Dx                             }
    Shl dx,1
    Shl dx,1                {Dx = Dx * 4                         }
    Add dx,ax               {Dx = Dx + Ax                        }
    Shl dx,1
    Shl dx,1
    Shl dx,1
    Shl dx,1
    Shl dx,1                {Dx = Dx * 16                        }
    Mov Cx,Dx               {Counter Cx sejumlah Dx              }
    Rep MovsW               {Pindahkan DS:SI ke ES:DI            }
    Pop ds                  {Pop Stack                           }
  End;
```

```
Procedure Put_Scroll_Page (Var PicBuff; Y:Integer);
  Begin
    If (Y<=Page_Width) and (Y>=1) then
      Begin
        If Page_Width-Y>=200 then
          MoveW(Page_Scroll[Y]^,PicBuff,32000)
        Else Begin
          MoveW(Page_Scroll[Y]^,PicBuff,(Page_Width-Y+1)*160);
          If Page_Width-Y+1<>200 then
         Put_Part(Page_Scroll[1]^,PicBuff,Page_Width-Y+1,200-(Page_Width-Y+1));
        End;
      End;
  End;

Procedure CloseVga256;
  Begin
    TextMode(80);
  End;

Begin
End.
```

Penjelasan Listing Program

Memang cukup membingungkan dengan adanya beberapa routine yang dibuat dengan bahasa Assembler. Tetapi jangan cepat putus asa, 'NO PAIN NO GAIN' (Tiada kemajuan tanpa diiringi dengan kesulitan). Berikut bahasan prinsip kerja dari unit di atas.

Unit ini dirancang sedemikian rupa untuk pembuatan game dan animasi profesional dengan sistem tiga layar. Hal-hal mengenai sistem tiga layar ini akan kita pelajari pada bab berikutnya. Untuk memberikan sedikit gambaran bagi Anda, sistem ini adalah teknik animasi dimana kita tidak langsung melakukan peletakan gambar pada layar monitor satu per-satu melainkan meletakannya terlebih dahulu pada virtual page (layar buatan yang tidak dapat kita lihat) yang secara cepat akan dipindahkan ke visual page (layar monitor). Karena itu maka setiap procedure yang akan meletakkan titik atau membaca titik pada layar akan diberi tambahan parameter alamat memori layar yang dituju, jadi tidak selalu ke visual page. Untuk sekarang ini, janganlah terlalu memikirkan teknik ini terlebih dahulu, nanti kita akan membahasnya habis-habisan.

STRUKTUR DATA

- Screen adalah konstanta yang berisi alamat visual page = $A000:0
- A,B adalah variabel counter.
- Reg adalah variabel register DOS.
- File_P adalah variabel file untuk proses pembacaan dan penulisan gambar.
- User_Pal adalah variabel tempat penyimpanan 256 warna palette ciptaan Anda.
- Default_Pal adalah variabel tempat penyimpanan 256 warna palette yang merupakan default (palette yang Anda install pada INSTALL_PALETTE).
- Active_Pal adalah variabel tempat penyimpanan 256 warna palette yang sedang aktif.
- Page_Scroll adalah variabel pointer di mana layar yang akan discroll disimpan.

INSTALASI MODE GRAPHIC 13h

Sebelum bermain-main di mode graphic 13h, tentunya kita harus menginstall screen ke mode graphic tersebut. Untuk itulah procedure-procedure di bawah ini dibuat.

PROCEDURE OPENVGA256;

Sintaks : OpenVga256;
Fungsi : Untuk menginstall Mode Graphic 13h

```
Procedure OpenVga256;
  Begin
    Reg.Ax:=$13;
    Intr($10,Reg);
  End;
```

Prosesnya yaitu dengan memanggil interrupt 10h servis 13h.

PROCEDURE CLOSEVGA256;

Sintaks : CloseVga256;
Fungsi : Untuk menutup Mode Graphic 13h

```
Procedure CloseVga256;
  Begin
    TextMode(80);
  End;
```

Prosesnya yaitu dengan memanggil statement TEXTMODE dari TURBO PASCAL dengan parameter width = 80.

PEMBUATAN BANGUN-BANGUN GEOMETRI

FASILITAS-FASILITAS PEMBUATAN BANGUN GEOMETRI

Procedure-procedure di bawah ini digunakan untuk pembuatan bangun-bangun geometri seperti LINE, RECTANGLE, CIRCLE dan lain-lainnya. Bangun-bangun tersebut tidak langsung digambar ke layar monitor (visual page) apabila Anda inginkan. Hal ini dilakukan agar kita dapat menerapkan sistem animasi tiga layar, di mana Anda dapat melakukan penggambaran pada virtual page (layar buatan yang tidak kita lihat). Apabila Anda ingin melakukan penggambaran langsung pada layar monitor, Anda dapat memasukkan parameter PicBuff sebagai alamat memori layar yaitu dengan SCREEN^. Pada awal unit VGACAD.TPU, SCREEN sudah dideklarasikan sebagai konstanta yang menunjukkan alamat memori layar = $A000:0.

PROCEDURE PUTPIXEL

Sintaks :	PutPixel (PicBuff^,X,Y,Col);		
Variabel :	PicBuff	=	Pointer Variabel yang berisi alamat dari PicBuff.
	X	=	Koordinat X dari titik.
	Y	=	Koordinat Y dari titik.
	Col	=	Warna dari titik.
Fungsi :	Untuk meletakkan titik pada memori page PicBuff.		

```
Procedure PutPixel (Var PicBuff; X,Y: Integer; Color: Byte);
ASSEMBLER;
  Asm
    Push ds      { Push Stack                                    }
```

Arahkan alamat ES:DI ke alamat PicBuff. PicBuff adalah memori page di mana kita akan menempatkan titik.

```
    Les di,PicBuff  {Arahkan ES:DI ke PICBUFF          }
```

Ambil koordinat Y dari titik. Kemudian dilakukan proses perhitungan offset dari titik tersebut. Proses ini sama dengan yang biasa kita lakukan dengan rumus Y*320. Pada program di bawah ini dilakukan optimasi kecepatan sehingga kita harus menghindari operasi perkalian dan pembagian (mengingat perkalian dan pembagian membutuhkan waktu jauh lebih lama dari penggeseran bit atau SHL dan SHR).

Maka untuk mengalikan Y dengan 320 dapat diganti dengan cara Offset=[(Y*4)+Y]*32. Untuk menghindari perkalian maka perkalian dengan 2 akan kita ganti dengan fungsi SHL dari assembler sehingga diperoleh algoritma di bawah ini.

```
Mov dx,Y        {Dx = Koordinat Y                }
Mov ax,dx       {Ax = Dx                         }
Shl dx,1        {Dx = Dx * 2                     }
Shl dx,1        {Dx = Dx * 2                     }
Add dx,ax       {Dx = Dx + Ax                    }
Shl dx,1
Shl dx,1
Shl dx,1
Shl dx,1
Shl dx,1
Shl dx,1        {Dx = Dx * 32                    }
```

Ambil koordinat X dari titik kemudian tambahkan ke alamat Offset. Mengingat rumus offset = Y*320+X.

```
Add dx,X                { Dx = Dx + Koordinat X }
```

Alamat offset dari titik sama dengan alamat offset yang kita hitung + offset dari memori page itu sendiri.

Di adalah offset dari memori page sedangkan Dx yang merupakan hasil perhitungan adalah alamat offset dari titik.

```
Add di,dx               { Offset = Offset  + Dx   }
```

Alokasi memori pada alamat ES:DI yang merupakan koordinat (X,Y) pada memori PicBuff dengan warna = Color.

```
  Mov al,Color      {Al = Warna titik              }
  Stosb             {Alokasi memori layar          }
  Pop ds            {Pop Stack                     }
End;
```

Sistem perhitungan offset di atas amat penting. Untuk itu disarankan agar Anda benar-benar menguasainya sebelum Anda meneruskan ke penjelasan selanjutnya. Sistem ini akan sering kita pergunakan pada setiap perhitungan posisi offset pada procedure-procedure VGACAD.TPU yang lainnya.

FUNCTION GETPIXEL

Sintaks : Col:=GetPixel (PicBuff^,X,Y);

Variabel : Col = Menerima warna yang dibaca.

Picbuff = Pointer variabel yang menerima alamat dari PicBuff.

X = Koordinat X.

Y = Koordinat Y.

Fungsi : Untuk membaca titik pada koordinat tertentu di memori page PicBuff.

```
Function  GetPixel (Var PicBuff;X,Y: Integer): Byte; ASSEMBLER;
  Asm
    Push ds        {Push Stack                                    }
```

Arahkan ES:DI ke pointer PicBuff dan hitung offset tempat titik akan dialokasikan pada memori page. Prosesnya sama dengan PUTPIXEL.

```
    Les di,PicBuff      {Arahkan ES:DI ke PICBUFF                 }
    Mov dx,Y            {Dx = Koordinat Y                         }
    Mov ax,dx           {Ax = Dx                                  }
    Shl dx,1
    Shl dx,1            {Dx = Dx * 4                              }
    Add dx,ax           {Dx = Dx + Ax                             }
    Shl dx,1
    Shl dx,1
    Shl dx,1
    Shl dx,1
    Shl dx,1
    Shl dx,1            {Dx = Dx * 32                             }
    Add dx,X            {Dx = Dx + Koordinat X                    }
    Add di,dx           {Offset = Offset  + Dx                    }
```

Baca nilai memori ES:DI dimana terletak titik yang akan dibaca ke dalam register AL. Register ini mewakili nilai output dari function GETPIXEL. sehingga function GETPIXEL akan berharga register AL.

```
    Mov al,[es:di]    {Pindahkan warna titik ke function      }
    Pop ds            {Pop Stack                              }
  End;
```

PROCEDURE RECTANGLE

Sintaks : Rectangle(PicBuff,X1,Y1,X2,Y2,Col);

Variabel :

PicBuff	=	Pointer variabel yang menerima alamat dari PicBuff.
X1,Y1,X2,Y2	=	koordinat kiri atas dan kanan bawah persegi panjang.
Col	=	Warna dari persegi panjang.

Fungsi : Untuk menggambar persegi panjang pada memori layar PicBuff.

```
Procedure Rectangle(Var PicBuff; X1,Y1,X2,Y2: integer; Col: Byte);
  Begin
```

Persegi panjang ini diciptakan dari penggabungan empat buah garis yang merupakan sisi-sisinya. Untuk itu dipanggil procedure LINE.

```
    Line(PicBuff,X1,Y1,X1,Y2,Col);
    Line(PicBuff,X1,Y2,X2,Y2,Col);
    Line(PicBuff,X2,Y2,X2,Y1,Col);
    Line(PicBuff,X2,Y1,X1,Y1,Col);
  End;
```

PROCEDURE BLOCK

Sintaks : Block (PicBuff^,X1,Y1,X2,Y2,Col);

Variabel :

PicBuff	=	Pointer variabel yang menerima alamat dari PicBuff.
X1,Y1,X2,Y2	=	koordinat kiri atas dan kanan bawah balok.
Col	=	Warna dari balok.

Fungsi : Untuk menggambar persegi panjang yang berisi warna pada memori layar PicBuff.

Pada procedure ini kita berhadapan dengan Assembler lagi dengan alasan optimasi kecepatan. Sebenarnya kitabisa memakai cara PUTPIXEL yang diulang secara horisontal dan vertikal dengan dua kali FOR. Cara ini sudah pernah kita gunakan pada program ANIM1.PAS

pada Bab 1 dan kecepatan yang dihasilkan kurang lebih 40 sampai 50 kali lebih lambat dari algoritma assembler pada procedure di bawah ini.

Sistem kerja dari Procedure BLOCK ini sebenarnya sama dengan cara pengulangan PUTPIXEL secara horisontal dan vertical. Bedanya, di sini kita tidak melakukan pemanggilan PUTPIXEL secara berulang kali yang berarti akan dilakukan perhitungan alamat offset dari titik setiap kali PUTPIXEL dipanggil. Bayangkan saja kalau balok yang akan kita buat sebesar 200 x 100 titik. PUTPIXEL akan dipanggil sejumlah 20000 kali yang berarti dilakukan perhitungan offset titik sejumlah itu pula. Justru di procedure PUTPIXEL sebagian besar kecepatan terkuras pada proses perhitungan offset. Maka pada procedure assembler ini, cukup dilakukan sekali saja perhitungan offset dari koordinat kiri atas balok. Misalnya kita hendak meletakkan balok pada koordinat (10,10) sampai (50,40). Program akan menghitung offset dari koordinat (10,10) yaitu 10*320 + 10 = 3210. Proses selanjutnya ialah mengadakan alokasi secara horisontal sejumlah panjang gambar = 50-10+1 dengan menggunakan REP STOSB karena titik-titik tersebut berada pada offset yang berurutan pada memori layar (Selisih 1 koordinat X sama dengan selisih 1 byte offset). Hal serupa tidak dapat kita terapkan secara vertical karena setiap selisih 1 koordinat Y sama dengan selisih 320 byte offset. Sehingga untuk penggambaran koordinat Y berikutnya, alamat alokasi offset harus dikembalikan ke 3210 (Mengingat setelah melakukan REP STOSB alokasi offset telah berpindah menjadi 3210 + panjang gambar = 3251). Kemudian dilakukan penambahan dengan 320 sehingga pada proses selanjutnya offset = 3210 + 320 = 3530. Proses ini diulang hingga seluruh balok telah dibentuk.

```
Procedure Block(Var PicBuff; X1,Y1,X2,Y2: Integer; Col: Byte);
ASSEMBLER;
  Asm
    Push ds              {Push Stack                          }
```

Arahkan DS:SI ke pointer PicBuff yang berfungsi sebagai memori layar.

```
    Les di,PicBuff       {Arahkan DS:SI ke PICBUFF      }
```

Hitung offset awal tempat penggambaran balok. Yaitu offset di mana kita akan mulai menggambar titik kiri atas dari balok. Tambahkan hasil perhitungan ke register DI menampung offset koordinat (0, 0) dari layar memori PicBuff.

```
Mov dx,Y1          {Dx = Y awal                    }
Mov ax,dx          {Ax = Dx                        }
Shl dx,1
Shl dx,1           {Dx = Dx * 4                    }
Add dx,ax          {Dx = Dx + Ax                   }
Shl dx,1
Shl dx,1
Shl dx,1
Shl dx,1
Shl dx,1
Shl dx,1           {Dx = Dx * 32                   }
Add dx,X1          {Dx = Dx + X awal               }
Add di,dx          {Offset = Offset + Dx           }
```

Hitung panjang dari balok yang akan digambar. Caranya yaitu dengan rumus X2-X1+1. Masukkan hasilnya ke dalam register BX.

```
Mov bx,X2          {Bx = X akhir                   }
Sub bx,X1          {Bx = Bx-X awal                 }
Inc bx             {Bx = Bx + 1 = Panjang Blok     }
```

Hitung lebar dari balok yang akan digambar. Caranya yaitu dengan rumus Y2-Y1+1. Masukkan hasilnya ke dalam register CX.

```
Mov cx,Y2          {Cx = Y akhir                   }
Sub cx,Y1          {Cx = Cx-Y awal                 }
Inc cx             {Cx = Cx + 1 = Lebar Blok       }
```

Isikan register Al dengan warna yang hendak kita pergunakan untuk membuat balok.

```
Mov Al,Col         {Al = Warna                     }
Cld                {Counter Offset Maju            }
@1:
```

Hitung lebar balok yang masih belum tergambar di layar.

```
Mov dx,cx          {Dx = Sisa lebar Blok           }
```

Isikan counter Cx dengan panjang balok. Counter ini berguna untuk menunjukkan pengulangan dari Rep StosB yang digunakan untuk mengalokasikan titik-titik secara horisontal.

```
Mov cx,bx          { Cx = Panjang Blok             }
```

Simpan terlebih dahulu nilai offset terakhir kali melakukan alokasi titik ke dalam register SI. Sehingga setelah melakukan alokasi secara horisontal kita dapat dengan mudah mengembalikan nilai SI ke nilai awal sebelum melakukan alokasi. Kemudian untuk alokasi horisontal koordinat Y berikutnya, tinggal menambahkan nilai SI tadi dengan 320.

```
Mov si,di       {Simpan  Si terakhir                }
Rep Stosb       {Alokasi titik sejumlah Cx          }
Mov di,si       {Kembalikan Si ke Si awal           }
Add di,320      {Si ke koordinat Y berikutnya       }
Mov cx,dx       {Cx = Dx                            }
```

Kurangi CX dengan 1. Register ini nantinya pada awal loop akan diisikan ke register DX yang berarti lebar gambar yang belum diletakkan berkurang 1. Loop akan keluar setelah CX berharga 0 yang berarti semua titik sepanjang lebar balok telah dialokasikan.

```
    Loop @1         {Cx-1, Ulang hingga Cx = 0          }
  Pop ds            {Pop Stack                          }
End;
```

PROCEDURE LINE

Sintaks : Line (PicBuff^,X1,Y1,X2,Y2,Col);

Variabel :

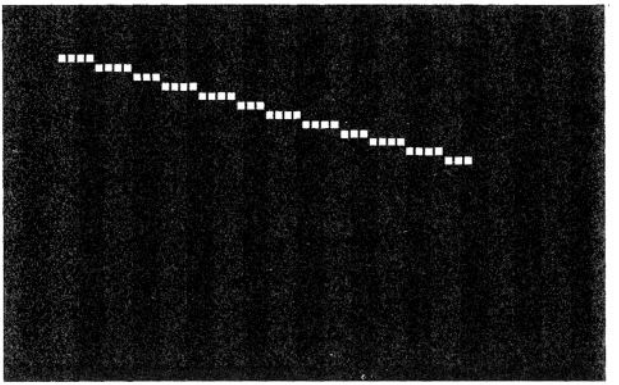
PicBuff	=	Pointer variabel yang menerima alamat dari PicBuff.
X1,Y1,X2,Y2	=	Koordinat awal dan akhir garis.
Col	=	Warna garis.

Fungsi : Berguna untuk membuat garis pada memori layar.

Screen Monitor adalah kumpulan titik-titik yang disusun secara horisontal-vertical. Satuan terkecil adalah pixel (titik). Kita tidak dapat mengisi separuh titik atau tiga perempat titik. Oleh karena itu pembuatan garis yang benar-benar lurus tidak mungkin kita lakukan secara sempurna.

Untuk itu dilakukan pembulatan ke titik yang terdekat apabila didapati koordinat pecahan. Misalnya kita hendak membuat garis dari koordinat (5,5) hingga (50,5), tidak ada masalah bukan? Karena garis tersebut adalah garis horisontal, kita tinggal melakukan looping PUTPIXEL dengan FOR secara horisontal, beres ! Tetapi bagaimana kalau garis yang hendak kita buat mulai koordinat (5,5) hingga (14,7).

Panjang horisontal dari garis adalah 10 titik dan panjang vertical dari garis adalah 3 titik. Dengan pembulatan maka setiap penambahan satu titik horisontal dilakukan penambahan titik vertical sejumlah 3/10 = 0,3. Apabila garis yang akan kita buat mulai koordinat (5,5) sampai (7,14) maka proses yang dilakukan akan dibalik karena sekarang panjang vertikal adalah 10 titik dan horisontal adalah 3 titik. Sehingga setiap penambahan satu titik vertikal dilakukan penambahan 3/10=0,3 titik. Procedure di bawah ini masih belum menggunakan Assembler karena dengan Assembler program ini akan sangat rumit tetapi berkecepatan puluhan kali lebih cepat. Bresenham Algorithm dengan Assembler akan kita bahas pada kesempatan yang akan datang di mana kita mungkin akan membuat game yang betul-betul rumit, misalnya game bertipe Wolfenstein 3D, Wing Commander ataupun DOOM sekalipun.



Gambar 3.1 Garis koordinat (5,5)-(51,17).

Pada Gambar 3.1, setiap dilakukan penambahan 1 pixel secara horisontal, dilakukan penambahan 3.8 pixel secara vertical. Karena sistem pixel tidak mengenal angka pecahan maka kadang-kadang 3.8 pixel dianggap 3 pixel atau 4 pixel. Algoritma pembuatan garis ini dikenal dengan nama Bresenham Algorithm. Procedure line berikut dibuat berdasarkan algoritma tersebut.

```
Procedure Line(Var PicBuff; X1,Y1,X2,Y2:integer; Col: Byte);
Var Dx,Dy,Loop,Ctr                    : Integer;
    XPos,YPos                         : Real;
    Rx,Ry                             : Real;
  Begin
```

Hitung panjang X dan panjang Y.

```
Dx:=X2-X1;
Dy:=Y2-Y1;
```

Isi koordinat Xpos dan Ypos adalah koordinat titik pertama dari garis.

```
XPos:=X1;YPos:=Y1;
```

Periksa apakah panjang Y dan X = 0. Karena hal ini bisa mengakibatkan Dy/Dx = tak terdefinisikan karena Dx = 0. Misalnya kita hendak membuat garis (10,10) sampai (10,10). Tentu saja Dx dan Dy = 0.

```
If Dx+Dy<>0 then
```

Periksa apakah panjang dari Dx lebih besar dari Dy ataukah sebaliknya. Ini dilakukan untuk mengetahui Dx atau Dy yang akan dibagi.

```
  If Abs(Dx)>Abs(Dy) then
    Begin
```

Penambahan koordinat X = 1.

```
      Rx:=1;
```

Penambahan koordinat Y = Dy/Dx.

```
      Ry:=Abs(Dy/Dx);
```

Jumlah pemanggilan PUTPIXEL adalah sejumlah Dx kali.

```
      Loop:=Abs(Dx);
  End Else
  Begin
```

Penambahan koordinat Y = 1.

```
      Ry:=1;
```

Penambahan koordinat X = Dx/Dy.

```
      Rx:=Abs(Dx/Dy);
```

Jumlah pemanggilan PUTPIXEL adalah sejumlah Dy kali.

```
    Loop:=Abs(Dy);
  End;
```

Jika panjang X berharga negatif maka arah penambahan koordinat X juga harus negatif. Ini berarti garis akan digambar dari kanan ke kiri. Misalnya saja Anda memanggil procedure ini dengan koordinat (50,10) sampai (10,30). Maka Dx = 10-50 = -40. Xpos dan Ypos yang merupakan titik start dari garis adalah 50 dan 10. Pada situasi ini Rx harus berharga negatif karena kita harus mengadakan peletakan titik mulai koordinat X = 50 hingga X = 10.

Demikian juga dengan penambahan koordinat Y perlu diproses pula dengan cara yang serupa.

```
If Dx<0 then Rx:=-Rx;
If Dy<0 then Ry:=-Ry;
```

Lakukan pemanggilan procedure PUTPIXEL untuk pertama kali.

```
PutPixel(PicBuff,Round(XPos),Round(YPos),Col);
```

Lakukan pemanggilan procedure PUTPIXEL untuk kelanjutan dari garis. Pada setiap loop dilakukan penambahan koordinat Xpos dan Ypos yang menunjukkan koordinat peletakan titik dengan procedure PUTPIXEL.

```
For Ctr:=1 to Loop do
  Begin
    XPos:=Xpos+Rx;
    YPos:=YPos+Ry;
    PutPixel(PicBuff,Round(XPos),Round(YPos),Col);
  End;
End;
```

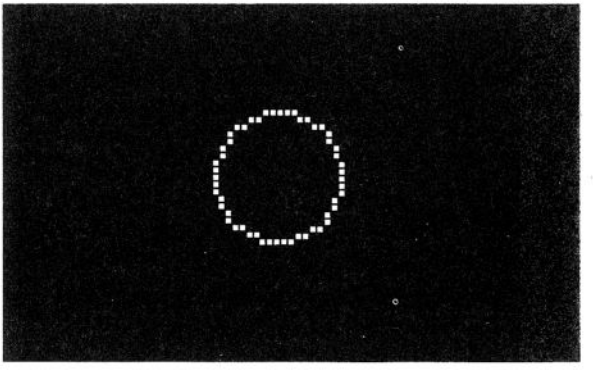
PROCEDURE CIRCLE

Sintaks : Circle (PicBuff^,X,Y,R,Col);
Variabel : PicBuff= Pointer variabel yang menerima alamat dari PicBuff.
X,Y = Koordinat titik pusat lingkaran.
R = Jaris-jari dari lingkaran.
Col = Warna dari lingkaran.
Fungsi : Untuk menggambar lingkaran pada memori layar.

Di sini kita gunakan fungsi persamaan parametrik lingkaran, di mana koordinat X didapat dari rumus R*Cos(Teta) dan koordinat Y = R*Sin(Teta). Teta adalah sudut mulai 0 hingga 359 derajat. Selanjutnya tinggal memanggil procedure PUTPIXEL pada koordinat tersebut.

```
Procedure Circle (Var PicBuff; X,Y: Integer; R: Integer; Col: Byte);
Var Teta: Integer;
  Begin
```

Variabel Teta akan di loop hingga 1440. Ini bukan berarti mulai 0 hingga 1440 derajat. Lihatlah pada pemanggilan procedure PUTPIXEL. Sudut yang dipergunakan pada Cos dan Sin adalah Teta*Pi/720. Ini berarti pada Teta = 1440, nilai yang dimasukkan ke dalam fungsi Cos dan Sin adalah 1440*Pi/720 = 2*Pi = 360 derajat. Kenapa kok tidak memakai For Teta:=0 to 360 saja? Alasannya ialah, apabila lingkaran semakin besar, dimana Radius semakin besar, dengan memakai interval 1 derajat lingkaran yang dihasilkan akan terputus-putus. Ini disebabkan karena keliling lingkaran besar sehingga selisih antara titik dengan titik berikutnya besar pula. Dengan memakai For Teta:=0 to 1440 berarti selisih antara titik tidak lagi 1 derajat melainkan 0.25 derajat sehingga jarak antara titik pada keliling lingkaran diperpendek. Sebenarnya ini bukanlah teknik yang baik sebagai algoritma pembuatan lingkaran, tetapi untuk saat ini kita masih tetap akan memakainya karena fungsi pembuatan lingkaran ini tidak banyak kita perlukan pada program-program selanjutnya.



Gambar 3.2 Lingkaran yang dihasilkan oleh procedure CIRCLE.

```
For Teta:=0 to 1440 do
   PutPixel(PicBuff,X+Round(R*Cos(Teta*pi/720)),Y+Round(R*Sin(Teta*pi/720)),Col)
  End;
```

PROCEDURE ELLIPSE

Sintaks : Ellipse (PicBuff,X,Y,Rx,Ry,Col);

Variabel :	PicBuff	=	Pointer variabel yang menerima alamat dari PicBuff.
	X,Y	=	Koordinat titik pusat ellips.
	Rx	=	Jari-jari horisontal dari ellips.
	Ry	=	Jari-jari vertical dari ellips.
	Col	=	Warna dari ellips.

Fungsi : Untuk membuat ellips pada memori layar.

Prinsip kerjanya sama dengan prinsip kerja lingkaran. Bedanya di sini terdapat dua buah jari-jari. Satu adalah jari-jari horisontal dan yang lain adalah jari-jari vertical. Hal ini disebabkan karena ellipse memiliki bentuk yang tidak simetris antara horisontal dan verticalnya. Masing-masing memiliki radius tersendiri. Sehingga persamaan parametrik dari ellipse untuk koordinat X = Rx*Cos(Teta) dan untuk koordinat Y = Ry*Sin(Teta).

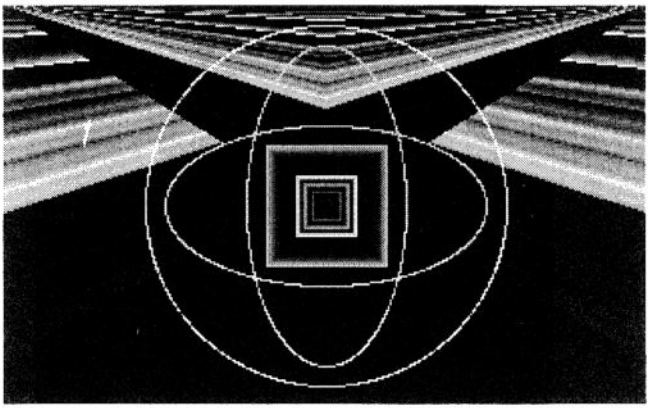
```
Procedure Ellipse (Var PicBuff; X,Y: Integer; Rx,Ry: Integer;
Col: Byte);
Var Teta: Integer;
  Begin
    For Teta:=0 to 1440 do

PutPixel(PicBuff,X+Round(Rx*Cos(Teta*pi/720)),Y+Round(Ry*Sin(Teta
*pi/720)),Col);
  End;
```

PRAKTEK PEMBUATAN BANGUN GEOMETRI DENGAN VGACAD.TPU

Setelah kita mengenal procedure-procedure di atas, alangkah baiknya kalau kita memanfaatkannya dengan mencoba membuat sebuah program. Perhatikan program di bawah ini.

Listing Program

```
{-------------------------------------------------------------------}
{File     :  Geometri1                                              }
{Compiler:  Turbo Pascal 7.00                                       }
{Tujuan   :  Untuk membuat gambar dengan fungsi geometri dari VGACAD.TPU }
{-------------------------------------------------------------------}

Uses Crt,Dos,VgaCad;
Var A:Integer;
Begin
  OpenVga256;
  For A:=1 to 199 do
    Begin
      Line(Screen^,0,0,319,A,A);
      Line(Screen^,319,0,0,A,A);
    End;
  For A:=1 to 30 do
    Rectangle(Screen^,160-A,100-A,160+A,100+A,A);
  Ellipse(Screen^,160,100,80,40,14);
  Ellipse(Screen^,160,100,40,80,14);
  Circle(Screen^,160,100,90,15);
  Readln;
  For B:=0 to 199 do
    For A:=0 to 319 do
      PutPixel(Screen^,A,B,0);
  CloseVga256;
End.
```



Gambar 3.1 Output Screen dari program GEOMETRI1.PAS

Penjelasan Listing Program

Tentunya Anda tidak menemukan kesulitan dalam mempelajari program ini. Struktur dasarnya mirip dengan program-program menggunakan GRAPH.TPU. Secara garis besar program ini akan menampilkan garis-garis berwarna dengan pola tirai kemudian dilanjutkan dengan pembuatan kotak berwarna-warna dengan pusat di tengah layar. Selanjutnya adalah penampilan elips dan lingkaran. Cobalah tekan Enter. Seluruh layar akan diisi dengan titik hitam mulai koordinat (0,0) hingga (319,199). Prosesnya hampir tidak terlihat karena sangat cepatnya. Di sinilah kelebihan yang kita dapatkan dari VGACAD.TPU.

Perhatikan pada setiap kali kita hendak membuat suatu bangun geometri. Pada awal parameter selalu terdapat SCREEN^. Ini berarti kita bertujuan untuk menggambar bangun tersebut langsung pada visual page. Karena SCREEN adalah konstanta Pointer = $A000:0 yang berarti alamat memori tempat monitor card mengadakan pembacaan (scanning).

ANIMASI TIGA LAYAR

Sistem ini hampir digunakan pada game-game yang beredar di pasaran. Dasarnya ialah dengan menyediakan dua buah *virtual page*. Virtual page adalah memori berjumlah 64000 bytes (sama dengan ukuran memori layar mode 13h yaitu $A000:0). Bagaimana cara membuat virtual page? Mudah, deklarasikan sebuah variabel pointer, kemudian lakukan GETMEM sejumlah 64000 bytes pada pointer tersebut. Anda dapat membuat virtual page sejumlah Anda suka tapi jumlah RAM yang dapat Anda pergunakan hanya 640 kbytes. Virtual page yang pertama kita namakan *background page* dan virtual page yang kedua ialah *process page*. Pada background page, kita letakkan gambar yang tidak akan kita ubah yaitu gambar background. Misalnya gambar arena pertempuran dengan pohon-pohon, lantai dan awan-awan. Sedangkan process page kita kosongkan. Kemudian pada saat animasi dijalankan, kita melakukan duplikasi gambar dari background page ke process page sehingga process page berisi gambar yang sama persis dengan background page. Setelah itu segala process peletakan gambar, seperti peletakan gambar orang yang sedang bertarung, kita lakukan pada process page ini. Langkah ketiga yaitu mengadakan duplikasi gambar dari process page ke visual page di mana kita dapat melihat. Proses ini dilakukan berulang-ulang sehingga menghasilkan animasi yang sangat halus dan tidak berkedip. Animasi yang dihasilkan tidak berkedip, tidak dilakukan proses penghapusan gambar secara langsung pada visual page sehingga kita tidak melihat gambar layar hitam setiap saat looping dilakukan, juga tidak dilakukan penghapusan gambar dengan cara penyimpanan gambar background di bawah sprite. Pada saat dilakukan proses duplikasi dari background page ke process page, kemudian dilanjutkan dengan peletakan gambar pada process page, gambar yang kita lihat di visual page adalah gambar yang dihasilkan pada looping sebelumnya. Dan karena akhirnya kita lakukan duplikasi process page ke visual page, otomatis gambar pada visual page akan berganti tanpa perlu kita hapus lagi. Kemudian ulangi lagi proses di atas.

Langkah-langkah proses animasi tiga layar ini adalah sebagai berikut:

- Pindahkan background page ke process page.

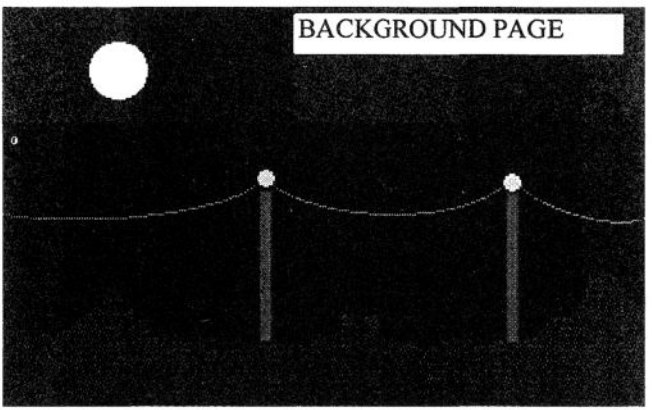
- Ubah koordinat dan kondisi sprite.
- Letakkan sprite pada process page.
- Pindahkan process page ke visual page.
- Ulang lagi ke proses awal.

Kita umpamakan kita sedang membuat animasi orang berlari di taman. Lihatlah penjelasan secara mendetil yang dilengkapi dengan gambar berikut ini.

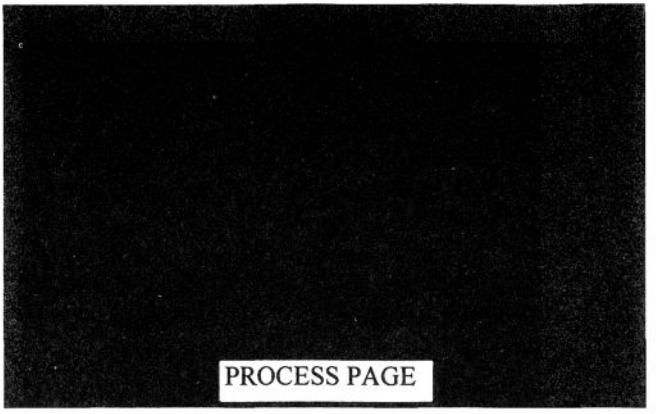
PERSIAPAN ANIMASI TIGA LAYAR

Proses yang dilakukan pada tahap persiapan ini cukup dilakukan pada awal program. Proses tersebut tidak perlu diikutsertakan dalam looping. Adapun proses-proses tersebut adalah sebagai berikut:

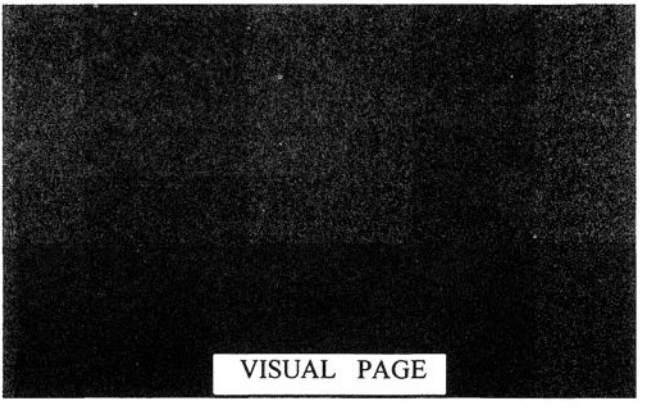
MENGISI BACKGROUND PAGE DENGAN GAMBAR TAMAN DAN KOSONGKAN PROCESS PAGE DAN VISUAL PAGE



Gambar 3. 2 Load gambar taman ke dalam background page

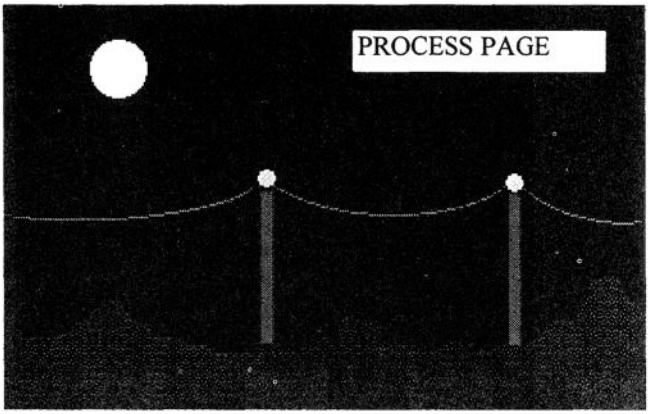


Gambar 3. 3 Pada saat PROCESS PAGE belum berisi apa-apa

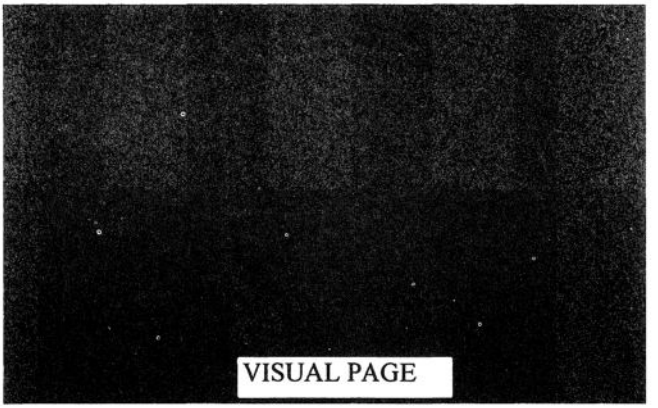


Gambar 3.4 VISUAL PAGE masih belum berisi apa-apa. Yang kita lihat adalah layar hitam.

MEMINDAHKAN BACKGROUND PAGE KE PROCESS PAGE

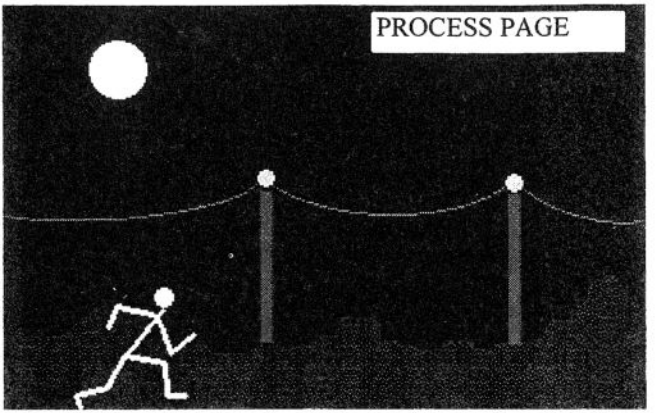


Gambar 3.5 PROCESS PAGE diisi dengan gambar taman yang disimpan pada BACKGROUND PAGE.

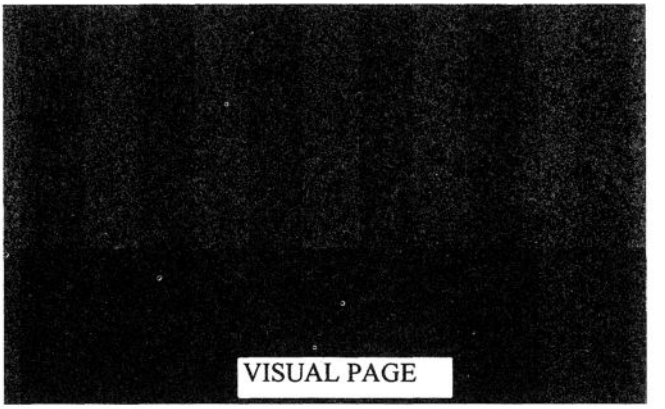


Gambar 3.6 VISUAL PAGE masih belum berisi apa-apa. Pada saat ini kita masih melihat layar hitam

MELETAKKAN SPRITE KE LAYAR PROCESS

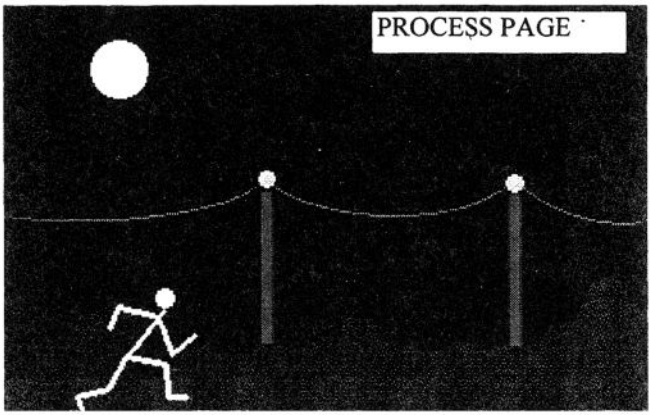


Gambar 3.7 Sprite orang berlari diletakkan di PROCESS PAGE

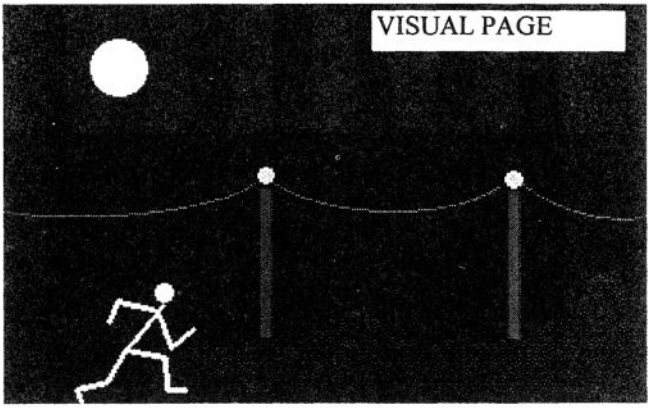


Gambar 3.8 VISUAL PAGE kita masih belum bisa melihat apa-apa karena segala proses masih kita lakukan pada PROCESS PAGE

MEMINDAHKAN PROCESS PAGE KE VISUAL PAGE



Gambar 3.9 PROCESS PAGE telah siap dengan gambar taman dengan sprite orang berlari. Tetapi PROCESS PAGE ini bukanlah layar yang kita lihat maka sebelum kita lakukan pemindahan PROCESS PAGE ke VISUAL PAGE, kita masih belum dapat melihat gambar ini



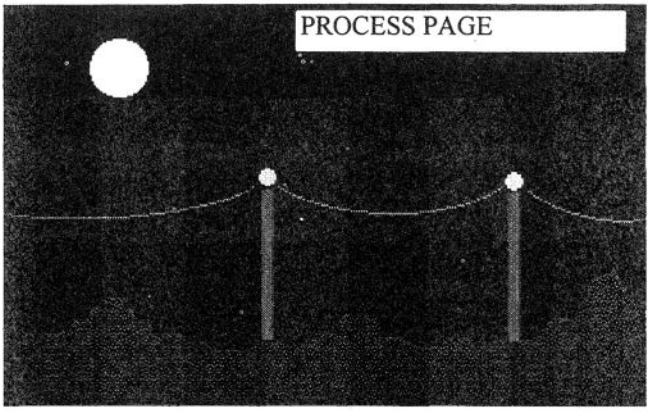
Gambar 3.10 Setelah PROCESS PAGE dipindahkan ke VISUAL PAGE, kita melihat gambar taman beserta sprite orang berlari

Sampai di sinilah tahap persiapan ANIMASI TIGA LAYAR berakhir. Tahap berikutnya adalah proses yang akan dilooping sepanjang animasi dimainkan.

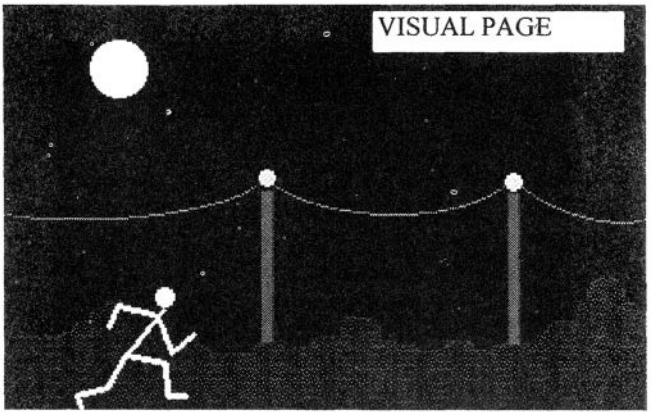
LOOPING ANIMASI TIGA LAYAR

Proses-proses dibawah inilah yang akan dilooping sepanjang animasi berlangsung. Adapun proses-proses tersebut adalah sebagai berikut:

MEMINDAHKAN BACKGROUND PAGE KE PROCESS PAGE

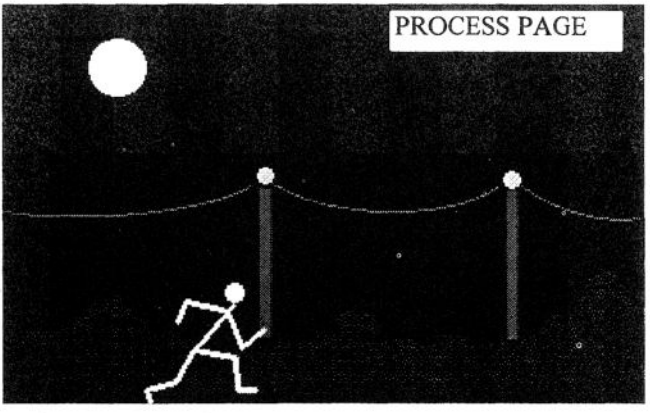


Gambar 3.11 Setelah dilakukan pemindahan BACKGROUND PAGE ke PROCESS PAGE, maka isi dari PROCESS PAGE ialah gambar taman kosong tanpa ada gambar orang berlari. Dengan cara ini kita tidak perlu repot-repot lagi menghapus sprite orang berlari yang ada di layar.

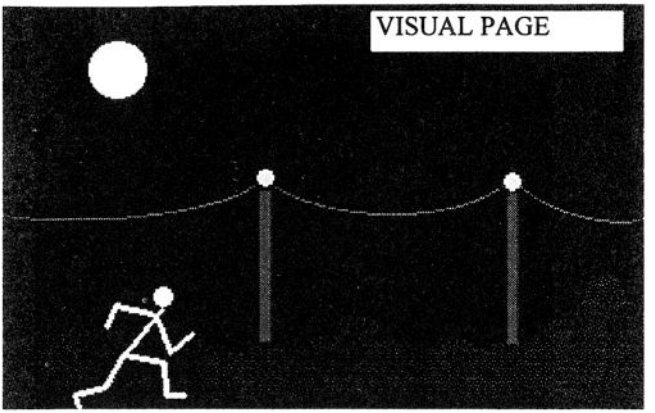


Gambar 3.12 Perubahan yang kita lakukan pada PROCESS PAGE tidak mempengaruhi VISUAL PAGE. Sehingga gambar yang sedang kita lihat adalah gambar taman dengan orang berlari pada posisi yang lama

MELETAKKAN SPRITE PADA PROCESS PAGE

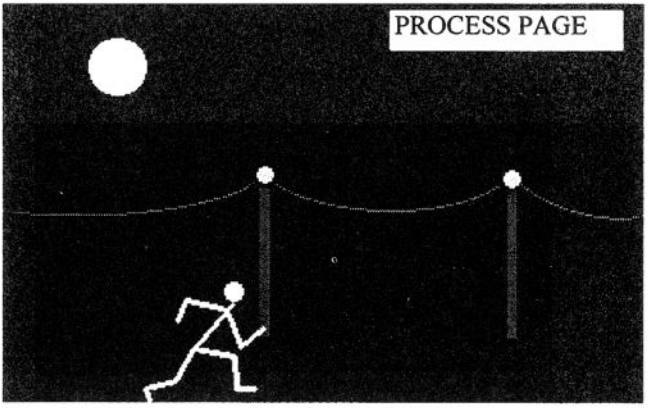


Gambar 3.13 Letakkan sprite orang berlari pada posisi yang baru. Proses peletakan ini tidak kita lihat karena dilakukan pada PROCESS PAGE.

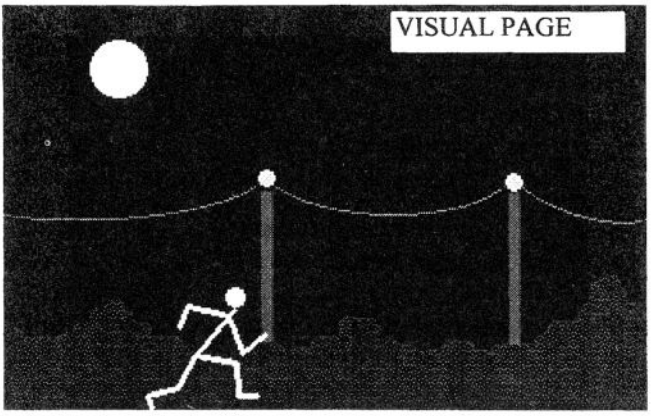


Gambar 3.14 Pada VISUAL PAGE kita masih belum melihat perubahan posisi gambar orang berlari karena VISUAL PAGE masih berisi gambar yang lama. Perubahan baru dapat kita lihat pada saat dilakukan pemindahan dari PROCESS PAGE ke VISUAL PAGE.

PINDAHKAN PROCESS PAGE KE VISUAL PAGE



Gambar 3.15 Pada PROCESS PAGE, telah tersedia gambar taman dengan gambar orang berlari pada posisi yang baru.



Gambar 3.16 Pada saat PROCESS PAGE dipindahkan ke VISUAL PAGE, gambar taman dengan gambar orang berlari pada posisi yang lama akan ditumpuk dengan gambar yang baru sehingga seakan-akan kita melihat gambar orang tersebut berpindah.

MENGULANG LAGI KE PROSES 'MEMINDAHKAN BACKGROUND KE PROCESS PAGE'

Untuk melakukan animasi tiga layar ini kita harus punya procedure-procedure untuk menampilkan gambar tidak hanya pada VISUAL PAGE saja melainkan pada setiap PAGE (seperti PROCESS PAGE dan BACKGROUND PAGE). Dengan alasan ini maka setiap procedure peletakan gambar ke memori layar (PutImage, Line, Circle dan sebagainya) pada unit VGACAD.TPU disertai dengan parameter alamat memori layar yang dituju.

FASILITAS-FASILITAS ANIMASI TIGA LAYAR

PROCEDURE DEFINE PAGE

Sintaks : Define_Page (PicBuff^);
Variabel : PicBuff = Pointer tempat virtual page akan dibuat.
Fungsi : Untuk menciptakan virtual page.

```
Procedure Define_Page(Var PicBuff: Pointer);
  Begin
    GetMem(PicBuff,64000);
  End;
```

PROCEDURE UNDEFINE_PAGE

Sintaks : UnDefine_Page(PicBuff^);
Variabel : PicBuff = Pointer alamat virtual page.
Fungsi : Untuk melenyapkan virtual page.

Procedure ini kita gunakan apabila virtual page tersebut sudah tidak kita pakai lagi.

```
Procedure UnDefine_Page(Var PicBuff: Pointer);
  Begin
    FreeMem(PicBuff,64000);
  End;
```

PROCEDURE FILL PAGE

Sintaks : Fill_Page(PicBuff^,Col);
Variabel : PicBuff = Pointer alamat memori layar yang hendak dihapus.
Fungsi : Untuk menghapus (bukan melenyapkan, tetapi cuma mengisi seluruh layar dengan warna tertentu) memori layar dengan warna tertentu.

FillWord yang dipanggil oleh procedure ini berfungsi untuk mengisi suatu nilai WORD pada alamat pointer PicBuff. Fungsinya sama dengan FILLCHAR dari Turbo Pascal, tetapi di sini dilakukan pemindahan secara WORD sehingga kecepatan yang diperoleh adalah dua kali lipat. Satu Word = 2 bytes. Sehingga jumlah word yang dipindahkan adalah 32000 words = 64000 bytes.

```
Procedure Fill_Page (Var PicBuff; Col: Byte);
  Begin
    FillWord(PicBuff,32000,Col);
  End;
```

PROCEDURE MOVE TO

Sintaks : MoveTo(Source^,Destination^);
Variabel : Source = Alamat page asal.
Destination = Alamat page tujuan.

Fungsi : Untuk menduplikatkan suatu virtual page ke virtual page yang lainnya. Ini digunakan pada saat kita hendak melakukan duplikasi dari background page ke process page. Juga pada saat menduplikasikan process page ke visual page.

Untuk itu dilakukan pemanggilan MOVEW yang berfungsi seperti MOVE nya Turbo Pascal tetapi di sini kita menggunakan pemindahan secara WORD.

```
Procedure MoveTo (Var Source, Destination);
  Begin
    MoveW(Source,Destination,32000);
  End;
```

FUNCTION IMAGE SIZE

Sintaks : Size:=ImageSize(X1,Y1,X2,Y2);
Variabel : X1,Y1,X2,Y2 = Koordinat kiri atas dan kanan bawah dari kotak di mana sprite akan direkam.
Fungsi : Untuk mengetahui jumlah memori yang diperlukan untuk merekam sprite ke memori dengan menggunakan GETIMAGE.
Di sini dipergunakan rumus:

Size = (Panjang Sprite * Lebar Sprite) + 4

Empat bytes pertama untuk menyimpan panjang dan lebar dari sprite (dua buah variabel integer = 4 bytes). Sisanya untuk menyimpan warna tiap titik dari sprite (1 titik = 256 warna = 1 bytes).

Tentunya Anda masih ingat rumus ini .(Telah dibahas pada bab I).

```
Function  ImageSize (X1,Y1,X2,Y2: Integer): Word;
  Begin
    ImageSize:= (X2-X1 + 1) * (Y2-Y1 + 1) + 4;
  End;
```

PROCEDURE PUT IMAGE

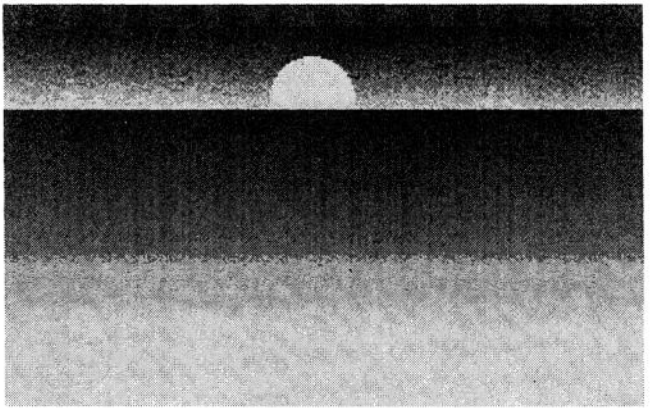
Sintaks : PutImage (PicBuff^,X,Y,Sprite^);
Variabel : PicBuff = Pointer yang menunjukkan screen page yang dituju.
X , Y = Koordinat (X,Y) tempat sprite akan diletakkan.
Fungsi : Untuk meletakkan sprite pada memori layar.

Pada procedure ini, sekali lagi diperlukan rutin Assembler karena PutImage inilah yang memegang peranan paling penting dalam animasi dan amat sering digunakan. Tanpa procedure ini tentunya kita tidak dapat meletakkan sprite. Untuk itulah kecepatan tinggi diperlukan. Maka mau tidak mau kita harus membuatnya dalam bahasa Assembler.

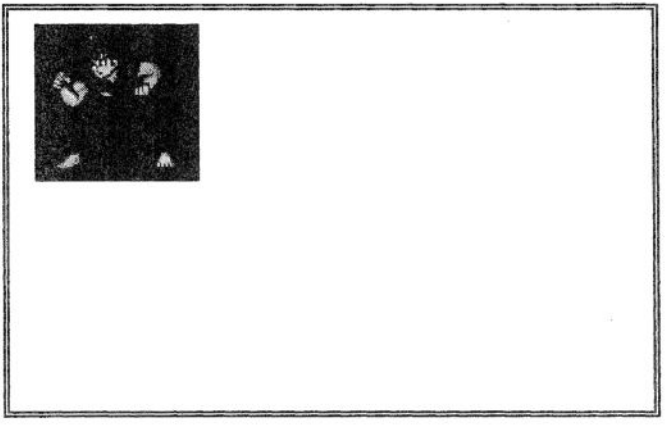
Terdapat beberapa perbedaan antara PutImage ini dengan PutImage yang disediakan oleh GRAPH.TPU. Di sini sprite dapat Anda letakkan baik di virtual page maupun di visual page tergantung dari parameter yang Anda inputkan pada PicBuff. Selain itu, kita tidak mengenal adanya XORPUT, ANDPUT dan lain-lainnya yang kurang begitu cocok untuk animasi. Tentunya Anda masih ingat bahwa dengan menggunakan PUTIMAGE dari GRAPH.TPU, kita tidak dapat menggerakkan sprite yang tidak berbentuk kotak dengan sempurna (Terjadi efek transparan, terjadi kekacauan pada gambar background dan lain-lainnya). Struktur data dari sprite hasil GETIMAGE nya VGACAD di memori sama persis dengan struktur data sprite yang dihasilkan oleh GETIMAGE dari GRAPH.TPU. Yaitu byte pertama dan kedua berisi panjang dari sprite (variabel integer memeerlukan 2 bytes). Byte ketiga dan keempat berisi lebar sprite. Sisanya adalah data dari tiap titik yang disimpan. Bentuk dari sprite yang disimpan juga berwujud kotak. Sehingga rumus total memori yang kita gunakan untuk menyimpan suatu sprite adalah:

```
Total_Memory = Panjang * Lebar + 4
```

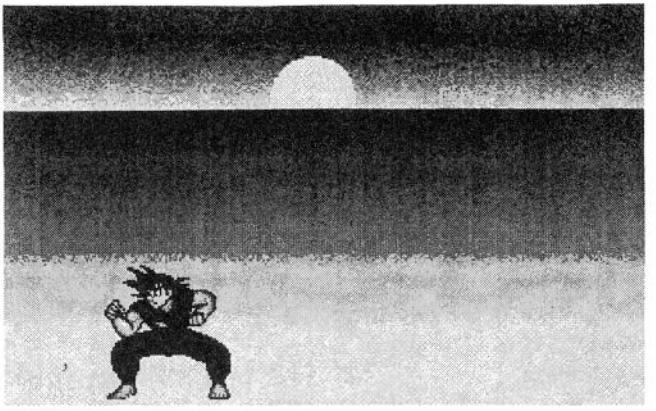
Perbedaan yang kedua adalah pada proses peletakan titik. Semua titik kecuali titik bernomor warna 0 (hitam) pada memori sprite akan diletakkan apa adanya. Maksudnya ialah bila warna titik tersebut warna nomor 4, titik yang diletakkan adalah warna nomor 4 juga. Bukannya seperti PUTIMAGE dari GRAPH.TPU yang melakukan operasi Boolean terhadap titik yang telah berada di layar. Kenapa kok titik hitam tidak diletakkan di layar? Dengan inilah maka peletakan gambar yang tidak berbentuk kotak dapat dilakukan. Jadi apabila Anda hendak menggerakkan gambar berbentuk bola. Buatlah dahulu gambar bola tersebut pada layar dengan background hitam. Kemudian lakukan GETIMAGE. Pada saat Anda melakukan PUTIMAGE, titik-titik di sekitar bola tidak akan ikut diletakkan pada layar karena bernomor warna 0 (hitam).



Gambar 3.17 Ini adalah gambar background dimana kita hendak meletakkan sprite.



Gambar 3.18 Sprite yang hendak kita letakkan pada gambar background yang disimpan di dalam memori adalah sejumlah panjang x lebar kotak hitam tersebut. Warna hitam memiliki kode 0 di dalam penyimpanan memori.



Gambar 3.19 Gambar sprite diletakkan pada gambar background. Warna yang memiliki kode 0 tidak diletakkan ke layar sehingga sprite yang kita letakkan benar-benar bagus dan tidak berbentuk kotak

```
Procedure PutImage (Var PicBuff;X,Y: Integer; Var Buffer);
ASSEMBLER;
  Asm
    Push ds             { Push Stack                        }
```

Arahkan ES:DI ke PicBuff yang merupakan memori layar yang dituju.

```
    Les di,PicBuff      { Arahkan ES:DI ke PICBUFF      }
```

Arahkan DS:SI ke Buffer yang merupakan memori tempat penyimpanan sprite.

```
    Lds si,Buffer       { Arahkan DS:SI ke BUFFER       }
```

Baca panjang dari sprite pada byte pertama dan kedua pada alamat memori DS:SI.

```
    Mov bx,[ds:si]      { Baca Panjang                      }
    Inc si              { Offset + 1                        }
    Inc si              { Offset + 1                        }
```

Baca lebar dari sprite pada byte ketiga dan keempat pada alamat memori DS:SI.

```
Mov cx,[ds:si]      { Baca Lebar                    }
Inc si              { Offset + 1                    }
Inc si              { Offset + 1                    }
```

Hitung offset awal di mana titik pertama dari sprite harus diletakkan. Ini dilakukan dengan menterjemahkan koordinat X dan Y pada parameter ke alamat offset. Caranya sama dengan proses yang dilakukan pada procedure PUPIXEL, GETPIXEL dan BLOCK terdahulu.

```
Mov dx,Y            { Masukkan Y ke Dx              }
Mov ax,dx           { Masukkan Dx ke Ax             }
Shl dx,1
Shl dx,1            { Dx = DX * 4                   }
Add dx,ax           { Dx = Dx + Ax                  }
Shl dx,1
Shl dx,1
Shl dx,1
Shl dx,1
Shl dx,1
Shl dx,1            { Dx = Dx * 32                  }
Add dx,X            { Dx = Dx + X                   }
Add di,dx           { Start Offset peletakan titik  }
```

Set counter DI dan SI maju.

```
Cld                 { Counter Offset Maju           }
@1:
```

Hitung lebar sprite yang masih belum ditampilkan pada memori layar.

```
Mov dx,cx           { Dx = Sisa Lebar Gambar        }
```

Isi register CX dengan panjang sprite yang akan ditampilkan di memori layar.

```
Mov cx,bx           { Cx = Sisa Panjang Gambar      }
```

Simpan alamat DI. Ini digunakan pada saat kita hendak memulai meletakkan titik pada koordinat Y berikutnya yang berarti alamat DI yang disimpan tersebut akan ditambah dengan 320.

```
Push di             { Push Stack                    }
@2:
```

Baca nomor warna titik dari sprite. Pada statement Lodsb, selain dilakukan pembacaan titik, dilakukan pula penambahan SI.

```
Lodsb                { Baca titik dari BUFFER          }
```

Periksa apakah titik tersebut berwarna 0 (hitam). Jika 0 lompat ke label @3 yang artinya tidak perlu dilakukan peletakan titik.

```
Or al,al             { Apakah titik = 0                }
Jz @3                { Kalau Y tidak perlu diletakkan  }
```

Lakukan peletakan titik pada memori layar. Statement Stosb otomatis akan melakukan penambahan register DI.

```
Stosb                { Letakkan titik                  }
```

Kurangi DI dengan 1 karena mau tidak mau program akan berlanjut ke label @3 di mana terdapat penambahan DI yang sebenarnya tidak diperlukan apabila peletakan titik dilakukan.

```
Dec di               { Kurangi offset pembacaan titik  }
@3:
  Inc di             { Tambah offset pembacaan titik   }
```

Ulangi lagi proses di atas sampai sudah dilakukan peletakan titik sejumlah panjang sprite pada memori layar.

```
Loop @2 {CX-1, Ulang sampai sisa panjang gambar CX = 0  }
```

Ambil lagi nilai DI yang telah disimpan. Ini dilakukan untuk proses penambahan offset ke koordinat Y berikutnya di mana DI harus dikembalikan ke posisi sebelum meletakkan titik secara horisontal. Setelah itu DI akan ditambah dengan 320.

```
Pop di               {Pop Stack                        }
Add di,320           {Register Di ke koordinat Y berikutnya  }
```

Ambil sisa lebar sprite yang belum diletakkan.

```
Mov cx,dx {Isi CX dengan sisa lebar   gambar           }

 Loop @1 { CX-1, Ulang hingga sisa lebar gambar  = 0   }
```

```
    Pop ds      { Pop Stack                                          }
  End;
```

PROCEDURE GET IMAGE

Sintaks : GetImage(PicBuff^,X1,Y1,X2,Y2,Sprite^);
Variabel : PicBuff = Screen Page tempat dilakukannya GetImage.
X1,Y1,X2,Y2 = Koordinat kotak di mana sprite akan diambil.
Sprite = Pointer alamat di mana sprite akan disimpan.
Fungsi : Untuk menciptakan sprite dengan gambar yang diambil dari memori layar.
Sistem kerja dari GetImage ini adalah sama dengan sistem kerja GetImage dari GRAPH.TPU.

```
Procedure GetImage (Var PicBuff; X1,Y1,X2,Y2: Integer; Var Buffer); ASSEMBLER;
  Asm
    Push ds  {Push Stack                                           }
```

Arahkan DS:SI ke PicBuff. PicBuff adalah tempat di mana kita akan mengambil sprite.

```
    Lds si,PicBuff { Arahkan DS:SI ke PICBUFF                      }
```

Arahkan ES:DI ke Buffer di mana sprite akan disimpan.

```
    Les di,Buffer      { Arahkan ES:DI ke BUFFER                   }
```

Hitung alamat offset kiri atas dari kotak sprite yang akan diambil.

```
    Mov dx,Y1          { Dx = Y awal                               }
    Mov ax,dx          { Ax = Dx                                   }
    Shl dx,1
    Shl dx,1           { Dx = Dx * 4                               }
    Add dx,ax          { Dx = Dx + Ax                              }
    Shl dx,1
    Shl dx,1
    Shl dx,1
    Shl dx,1
    Shl dx,1
    Shl dx,1           { Dx = Dx * 32                              }
    Add dx,X1          { Dx = Dx + X awal                          }
    Add si,dx          { Offset = Offset + Dx                      }
```

Hitung panjang sprite yang akan diambil. Caranya yaitu dengan rumus X2-X1+1. Kemudian masukkan ke byte pertama dan kedua pada memori di mana sprite akan disimpan.

```
Mov bx,X2          { Bx = X akhir                          }
Sub bx,X1          { Bx = Bx-X awal                        }
Inc bx             { Bx = Bx + 1 = Panjang Gambar          }
Mov [es:di],bx     { Tulis Panjang Gambar                  }
```

Pindahkan DI ke offset ketiga.

```
Inc di                { Offset = Offset + 1                }
Inc di                { Offset = Offset + 1                }
```

Hitung lebar dari sprite yang akan diambil. Caranya yaitu dengan rumus Y2-Y1+1. Kemudian masukkan ke byte ketiga dan keempat pada memori di mana sprite akan disimpan.

```
Mov cx,Y2          { Cx = Y akhir                          }
Sub cx,Y1          { Cx = Cx-Y awal                        }
Inc cx             { Cx = Cx + 1 = Lebar Gambar            }
Mov [es:di],cx     { Tulis Lebar Gambar                    }
```

Pindahkan DI ke offset kelima untuk mulai memasukkan data warna tiap titik.

```
Inc di             { Offset = Offset + 1                   }
Inc di             { Offset = Offset + 1                   }
```

Counter DI dan SI maju.

```
Cld                { Counter Offset Maju                   }
@1:
```

Hitung sisa lebar sprite yang belum direkam.

```
Mov dx,cx          { Dx = Sisa lebar gambar                }
```

Isikan register CX dengan panjang sprite untuk melakukan perekaman sejumlah panjang sprite dengan menggunakan Rep Movsb.

```
Mov cx,bx          { Cx = Panjang gambar                   }
```

Simpan dahulu register SI. Karena nantinya SI harus dikembalikan ke nilai pada saat ini untuk mulai merekam titik pada koordinat Y berikutnya. Pada proses selanjutnya akan dilakukan penambahan 320 pada register SI.

```
Mov ax,si          { Ax = Si                                    }
```

Lakukan perekaman titik dari DS:SI ke ES:DI sejumlah Cx kali. Setiap kali melakukan Mov dilakukan penambahan SI dan DI.

```
Rep Movsb          { Baca titik sejumlah Cx                     }
```

Kembalikan nilai register SI untuk melakukan penambahan dengan 320 yang berarti mulai merekam koordinat Y berikutnya.

```
Mov si,ax          { Kembalikan Si ke Si awal                   }
Add si,320         { Si ke koordinat Y berikutnya               }
```

Pindahkan jumlah lebar sprite yang belum direkam ke register CX.

```
Mov cx,dx          { Cx = Dx                                    }
```

Cx dikurangi 1 (Lebar sprite yang tersisa dikurangi 1). Ulang lagi ke atas hingga Cx = 0 yang berarti sudah tidak ada lagi titik yang perlu direkam.

```
    Loop @1        { Cx-1, Ulang hingga Cx = 0                  }
  Pop ds           { Pop Stack                                  }
End;
```

PROCEDURE LOAD PICTURE

Sintaks : Load_Picture(Nama_File,PicBuff^);
Variabel : Nama_File = Variabel STRING yang berisi nama dari gambar yang hendak Anda load.
Sprite = Screen Page tempat di mana gambar akan ditampilkan.
Fungsi : Untuk meload gambar ke Screen Page.

Pada pembuatan game-game, tentu pembuatan gambar background tidak dilakukan secara matematis. Maksudnya ialah tidak mungkin Anda merancang background dengan menggunakan procedure LINE, RECTANGLE, CIRCLE dan lain-lainnya. Bisa pun pasti rumit dan memakan waktu yang terlalu lama. Bayangkan saja Anda

harus menggambar gunung-gunung dan pepohonan dengan menggunakan fasilitas-fasilitas tersebut. Untuk itulah disediakan procedure untuk meload gambar dari disk. Gambar ini bisa Anda hasilkan sendiri dengan merancangnya pada visual page, kemudian gunakan procedure SAVE_PICTURE. Tapi hal ini tidaklah jauh berbeda dengan yang cara pertama karena otomatis Anda juga harus menggambar background tersebut terlebih dahulu.

Gambar yang hendak Anda load kompatibel dengan memori layar. Yaitu berjumlah 64000 bytes. Biasanya gambar ini dapat Anda hasilkan dengan bantuan software lain seperti Deluxe Paint, Photo Styler dan lain sebagainya yang nantinya Anda capture (tangkap) dari layar monitor dan menyimpannya ke dalam file. Untuk itu diperlukan suatu program resident yang akan aktif di memori (semacam virus) dan merekam alamat visual page = $A000:0 ke dalam file. Pembuatan program ini akan dibahas pada bab-bab berikutnya.

```
Procedure Load_Picture (File_Name:String;Var Buffer);
  Begin
    Assign(File_P,File_Name);
    Reset(File_P,1);
```

Baca gambar dari File dan masukkan ke memori layar Buffer.

```
    BlockRead(File_P,Buffer,64000);
    Close(File_P);
  End;
```

PROCEDURE SAVE PICTURE

Sintaks : Save_Picture(Nama_File,PicBuff^);
Variabel : Nama_File = Variabel STRING di mana berisi nama dari gambar yang hendak Anda save.
Sprite = Memori layar tempat gambar akan disimpan.
Fungsi : Untuk merekam gambar dari memori layar ke dalam file.

Dengan ini Anda dapat merekam apa yang Anda buat di memori layar ke dalam file sehingga sewaktu-waktu Anda tidak perlu membuatnya lagi melainkan tinggal meloadnya dari file yang telah Anda buat.

```
Procedure Save_Picture (File_Name:String;Var Buffer);
  Begin
    Assign(File_P,File_Name);
```

```
    Rewrite(File_P,1);
```

Rekam dari memori layar Buffer ke file.

```
    BlockWrite(File_P,Buffer,64000);
    Close(File_P);
  End;
```

PROCEDURE LOAD SPRITE

Sintaks : Load_Sprite (File_Name;Sprite;Var Size:Word);
Variabel : File_Name = Nama file sprite.
Sprite = Pointer di mana sprite akan diload.
Size = Besar sprite yang dibaca dari file.
Fungsi : Untuk membaca sprite yang telah disimpan di file.

Dengan adanya procedure ini maka kita tidak perlu setiap kali melakukan GETIMAGE pada memori layar setiap hendak menciptakan sprite. Untuk itu kita harus membuat sprite file. Caranya yaitu bisa dengan melakukan penangkapan gambar dengan menggunakan program CAPTURE. Juga bisa dengan menggunakan procedure SAVESPRITE.

```
Procedure Load_Sprite (File_Name:String;Var Ptr:Pointer; Var
Size:Word);
  Begin
    Assign(File_P,File_Name);
    Reset(File_P,1);
```

Pesan memori untuk penyimpanan sprite pada pointer PTR sebesar ukuran file yang akan dibaca.

```
    GetMem(Ptr,FileSize(File_P));
```

Baca file ke dalam PTR.

```
    BlockRead(File_P,Ptr^,FileSize(File_P));
```

Keluarkan jumlah memori yang diperlukan untuk merekam sprite sebagai output parameter.

```
    Size:=FileSize(File_P);
    Close(File_P);
  End;
```

PROCEDURE SAVE SPRITE

Sintaks : Save_Sprite (File_Name,Size);
Variabel : File_Name = Name file di mana sprite akan dimasukkan.
Size = Jumlah memori yang diperlukan untuk merekam sprite.
Fungsi : Untuk merekam sprite ke dalam file. Tujuannya apabila sewaktu-waktu kita hendak mempergunakan sprite tersebut kita tidak perlu melakukan GETIMAGE lagi dari memori layar.

```
Procedure Save_Sprite (File_Name:String;Ptr:Pointer;Size:Word);
  Begin
    Assign(File_P,File_Name);
    Rewrite(File_P,1);
```

Rekam sprite ke dalam file.

```
    BlockWrite(File_P,Ptr^,Size);
    Close(File_P);
  End;
```

PRAKTEK ANIMASI TIGA LAYAR DENGAN VGACAD.TPU.

Sampai sini kita sudah bisa menerapkan ilmu animasi tiga layar pada program. Sekarang lihatlah program di bawah ini. Di sini Anda akan menemukan betapa handalnya sistem animasi tiga layar.

Listing Program

```
{-------------------------------------------------------------}
{File     : 3Scr1.Pas                                         }
{Compiler: Turbo Pascal 7.00                                  }
{Fungsi   : Menggerakkan sprite dengan teknik animasi tiga layar.}
{-------------------------------------------------------------}

Uses Crt,Dos,VgaCad;
Const
    Up    = 1;
    Down  = 2;
    Left  = 3;
    Right= 4;
```

```
Var Back,Process: Pointer;
    Panah      : Pointer;
    A,B        : Integer;
    Dir        : Byte;
    Ch         : Char;
    Luar       : Boolean;

Procedure Initialize;
  Begin
    Define_Page(Back);
    Define_Page(Process);
    A:=160;B:=100;
    Luar:=False;
  End;

Procedure Generate_Picture;
  Begin
    For A:=0 to 99 do
      Begin
        Rectangle(Screen^,160-A,100-A,160+A,100+A,16+A);
      End;
    Readln;
    MoveTo(Screen^,Back^);
    Fill_Page(Screen^,0);
    Block(Screen^,158,80,163,120,14);
    Block(Screen^,140,98,180,103,14);
    Rectangle(Screen^,158,80,163,120,1);
    Rectangle(Screen^,140,98,180,103,1);
    GetMem(Panah,ImageSize(140,80,180,120));
    GetImage(Screen^,140,80,180,120,Panah^);
    Readln;
    Fill_Page(Screen^,0);
  End;

Procedure Main;
  Begin
    Repeat
      Repeat
        MoveTo(Back^,Process^);
        Case Dir of
          Up:If B>0 then Dec(B,1);
          Down:If B<150 then Inc(B,1);
          Left:If A>0 then Dec(A,1);
          Right:If A<270 then Inc(A,1);
        End;
        PutImage(Process^,A,B,Panah^);
        MoveTo(Process^,Screen^);
      Until Keypressed;
      Ch:=Readkey;
      Case Ch of
```

```
        #80:Dir:=Down;
        #72:Dir:=Up;
        #75:Dir:=Left;
        #77:Dir:=Right;
        #27:Luar:=True;
        ' ':Dir:=0;
      End;
    Until Luar;
  End;
Begin
  OpenVga256;
  Initialize;
  Generate_Picture;
  Main;
  CloseVga256;
End.
```

Penjelasan Listing Program

Pada saat program ini dijalankan Anda akan melihat kotak-kotak berwarna. Inilah yang nantinya merupakan background dari animasi. Tekanlah enter. Pada layar akan digambar tanda tambah. Inilah yang menjadi sprite yang akan kita gerakkan nanti. Tekanlah enter sekali lagi. Sekarang Anda akan melihat tanda tambah tersebut berada di atas background. Gerakkan tanda tambah tersebut dengan menekan tombol kursor (cukup sekali tekan saja maka sprite akan terus bergerak ke arah yang Anda pilih). Tekan SpaceBar untuk menghentikan gerakan sprite dan tekan ESC untuk keluar dari program. Bagaiman pendapat Anda mengenai animasi yang dihasilkan? Tidak ada efek transparant! Tidak ada satu pun yang berkedip! Kecepatan animasi tinggi! Gambar yang kita gerakkan tidak berbentuk kotak! Inilah yang kita inginkan untuk pembuatan game.

STRUKTUR VARIABEL

- Panah = Pointer tempat sprite akan disimpan.
- A,B = Koordinat dari sprite.
- Dir = Variabel yang menunjukkan instruksi yang harus dilaksanakan.
- Ch = Variabel karakter yang menunjukkan tombol yang ditekan.
- Luar = Variabel boolean untuk keluar dari loop.

PENJELASAN PROGRAM

Seperti biasa kita lakukan Instalasi ke mode 13h dengan menggunakan OPENVGA256.

```
Begin
  OpenVga256;
```

Panggil procedure INITIALIZE. Di sini dilakukan proses pembuatan Background dan Process Page.

```
Procedure Initialize;
  Begin
```

Ciptakan Backgrund Page yaitu pada alamat pointer Back.

```
    Define_Page(Back);
```

Ciptakan Process Page yaitu pada alamat pointer Process.

```
    Define_Page(Process);
```

Koordinat awal dari sprite = (160,100).

```
    A:=160;B:=100;
```

Keluar Loop = False.

```
    Luar:=False;
  End;
```

Proses selanjutnya ialah menggambar layar background dan sprite. Untuk itu dipanggil procedure GENERATE_PICTURE.

```
Procedure Generate_Picture;
  Begin
```

Gambar background di visual page.

```
    For A:=0 to 99 do
      Begin
        Rectangle(Screen^,160-A,100-A,160+A,100+A,16+A);
      End;
    Readln;
```

Pindahkan background dari visual page ke background page (Back).

```
MoveTo(Screen^,Back^);
```

Hapus visual page.

```
Fill_Page(Screen^,0);
```

Gambar tanda tambah pada visual page sebagai sprite.

```
Block(Screen^,158,80,163,120,14);
Block(Screen^,140,98,180,103,14);
Rectangle(Screen^,158,80,163,120,1);
Rectangle(Screen^,140,98,180,103,1);
```

Ambil tanda tambah sebagai sprite. Untuk itu digunakan variabel Panah. Sebelumnya dilakukan pemesanan memori sejumlah IMAGESIZE(140,80,180,120).

```
GetMem(Panah,ImageSize(140,80,180,120));
GetImage(Screen^,140,80,180,120,Panah^);
Readln;
```

Hapus visual page.

```
    Fill_Page(Screen^,0);
  End;
```

Panggil procedure MAIN.

```
Procedure Main;
  Begin
    Repeat
      Repeat
```

Duplikasi background page ke process page.

```
        MoveTo(Back^,Process^);
```

Pindahkan Koordinat Sprite.

```
        Case Dir of
          Up:If B>0 then Dec(B,1);
```

```
        Down:If B<150 then Inc(B,1);
        Left:If A>0 then Dec(A,1);
        Right:If A<270 then Inc(A,1);
      End;
```

Meletakkan sprite pada process page.

```
      PutImage(Process^,A,B,Panah^);
```

Menduplikasi process page ke visual page. Di sini gambar sudah bisa kita lihat.

```
      MoveTo(Process^,Screen^);
    Until Keypressed;
```

Mendeteksi tombol yang ditekan dan memasukkan instruksi ke dalam variabel DIR.

```
    Ch:=Readkey;
    Case Ch of
      #80:Dir:=Down;
      #72:Dir:=Up;
      #75:Dir:=Left;
      #77:Dir:=Right;
      #27:Luar:=True;
      ' ':Dir:=0;
    End;
  Until Luar;
End;
```

Struktur program pada procedure MAIN hampir sama dengan struktur program animasi kita yang terdahulu bukan? Bedanya pada looping kedua digunakan sistem animasi tiga layar, yaitu:

- Pindahkan background page ke process page.
- Ubah koordinat dan kondisi sprite.
- Letakkan sprite pada Process Page.
- Pindahkan Process Page ke Visual Page.
- Ulang lagi ke proses awal.

Langkah terakhir ialah menutup mode graphic 13h yaitu dengan pemanggilan procedure CLOSEVGA256.

```
CloseVga256;
End.
```

Program sederhana di atas telah menunjukkan bagaimana mempergunakan fasilitas animasi tiga layar dari VGACAD.TPU. Tentunya kita tidak ingin selalu menggambar sprite terlebih dahulu pada awal program. Lebih baik sprite itu kita load dari file sehingga program kita menjadi lebih profesional. Untuk itulah maka disediakan procedure SAVE_SPRITE dan LOAD_SPRITE. Anda dapat menciptakan bermacam-macam sprite kemudian menyimpannya dengan menggunakan SAVE_SPRITE. Nantinya pada program-program Anda selanjutnya, apabila Anda ingin mempergunakan sprite yang sama, Anda tidak perlu lagi menciptakannya dari awal. Cukup dengan memanggil sprite yang telah Anda simpan berupa file. Untuk lebih jelasnya cobalah kedua program di bawah ini.

Listing Program

```
{-------------------------------------------------------------}
{ File     :  3Scr2.Pas                                       }
{ Compiler :  Turbo Pascal 7.00                               }
{ Fungsi   :  Untuk menciptakan file sprite panah.            }
{-------------------------------------------------------------}

Uses Crt,Dos,VgaCad;

Var Size:Word;
    Panah:Pointer;

Begin
  OpenVga256;
  Block(Screen^,158,80,163,120,14);
  Block(Screen^,140,98,180,103,14);
  Rectangle(Screen^,158,80,163,120,1);
  Rectangle(Screen^,140,98,180,103,1);
  Size:=ImageSize(140,80,180,120);
  GetMem(Panah,Size);
  GetImage(Screen^,140,80,180,120,Panah^);
  Save_Sprite('Panah.Spr',Panah,Size);
  CloseVga256;
End.
```

Penjelasan Listing Program

Program ini berguna untuk menciptakan sebuah sprite yaitu tanda tambah. Kemudian tanda tambah tersebut akan dimasukkan ke dalam file melalui procedure SAVE_SPRITE dengan nama file Panah.Spr.

PENJELASAN PROGRAM

Ciptakan gambar tanda tambah pada visual page.

```
Block(Screen^,158,80,163,120,14);
Block(Screen^,140,98,180,103,14);
Rectangle(Screen^,158,80,163,120,1);
Rectangle(Screen^,140,98,180,103,1);
```

Kalkulasi jumlah memori yang dibutuhkan untuk menyimpan sprite tersebut ke dalam memori.

```
Size:=ImageSize(140,80,180,120);
```

Pesan memori sejumlah Size.

```
GetMem(Panah,Size);
```

Ambil sprite dari visual page ke dalam alamat pointer Panah.

```
GetImage(Screen^,140,80,180,120,Panah^);
```

Masukkan sprite dengan ukuran size ke dalam file Panah.Spr.

```
Save_Sprite('Panah.Spr',Panah,Size);
```

Setelah program ini dijalankan Anda akan mendapati file Panah.Spr yang berisi sprite yang disimpan.

Listing Program

```
{------------------------------------------------------------ }
{ File     :  3Scr3.Pas                                       }
{ Compiler :  Turbo Pascal 7.00                               }
{ Fungsi   :  Untuk menggerakkan sprite yang dibaca dari file. }
{------------------------------------------------------------ }
```

```
Uses Crt,Dos,VgaCad;
Const
    Up    = 1;
    Down  = 2;
    Left  = 3;
    Right= 4;

Var Back,Process: Pointer;
    Panah      : Pointer;
    Size       : Word;
    A,B        : Integer;
    Dir        : Byte;
    Ch         : Char;
    Luar       : Boolean;

Procedure Initialize;
  Begin
    Define_Page(Back);
    Define_Page(Process);
    A:=160;B:=100;
    Luar:=False;
  End;
Procedure Generate_Picture;
  Begin
    For A:=0 to 99 do
      Begin
        Rectangle(Screen^,160-A,100-A,160+A,100+A,16+A);
      End;
    Readln;
    MoveTo(Screen^,Back^);
    Fill_Page(Screen^,0);
    Load_Sprite('Panah.Spr',Panah,Size);
  End;

Procedure Main;
  Begin
    Repeat
      Repeat
        MoveTo(Back^,Process^);
        Case Dir of
          Up:If B>0 then Dec(B,1);
          Down:If B<150 then Inc(B,1);
          Left:If A>0 then Dec(A,1);
          Right:If A<270 then Inc(A,1);
        End;
        PutImage(Process^,A,B,Panah^);
        MoveTo(Process^,Screen^);
      Until Keypressed;
      Ch:=Readkey;
```

```
      Case Ch of
        #80:Dir:=Down;
        #72:Dir:=Up;
        #75:Dir:=Left;
        #77:Dir:=Right;
        #27:Luar:=True;
        ' ':Dir:=0;
      End;
    Until Luar;
  End;

Begin
  OpenVga256;
  Initialize;
  Generate_Picture;
  Main;
  CloseVga256;
End.
```

Penjelasan Listing Program

Program ini sama dengan program 3SCR1.PAS di atas. Hanya saja, di sini kita tidak lagi melakukan penggambaran tanda tambah sebagai sprite, melainkan tinggal meload-nya dari file yang telah dibuat oleh program 3SCR2.PAS. Coba lihat procedure GENERATE_PICTURE.

```
Procedure Generate_Picture;
  Begin
    For A:=0 to 99 do
      Begin
        Rectangle(Screen^,160-A,100-A,160+A,100+A,16+A);
      End;
    Readln;
    MoveTo(Screen^,Back^);
    Fill_Page(Screen^,0);
```

Dilakukan pembacaan sprite dari file PANAH.SPR. Size adalah output variabel yang mengeluarkan besar memori yang dibutuhkan oleh sprite. Perlu diingat bahwa untuk melakukan load sprite Anda tidak perlu melakukan pemesanan memori pada variabel pointer Panah. Proses ini secara otomatis akan dilakukan oleh procedure LOAD_SPRITE.

```
    Load_Sprite('Panah.Spr',Panah,Size);
  End;
```

Sekarang Anda telah menguasai teknik animasi tiga layar ini. Cobalah mencoba membuat program-program animasi dengan teknik ini. Ingatlah, untuk membuat game yang bagus Anda harus cukup pengalaman dan sering mencoba.

PENGOLAHAN GAMBAR

Hingga saat ini kita telah dapat membuat animasi sprite bergerak tanpa adanya masalah yang perlu dikuatirkan seperti efek transparan, sprite berkedip dan lain sebagainya. Tetapi ada sesuatu yang masih belum cocok. Mungkin Anda kesulitan dalam pembuatan sprite. Bagaimana cara membuat sprite bergambar orang? Apakah kita harus merancangnya satu per satu menggunakan LINE, CIRCLE, PUTPIXEL dan lain sebagainya? VGACAD.TPU tidak sejelek yang Anda bayangkan. Masalah ini dapat diatasi dengan fasilitas pengolahan gambar pada unit VGACAD.TPU Di sini kita akan mempelajari sistem kerja program capture.

Tentunya Anda semua tahu program INSET Word Star. Program ini digunakan untuk menangkap gambar yang ada pada layar monitor dan menyimpannya ke dalam suatu file. Tujuan sebenarnya ialah untuk memanfaatkan gambar yang dihasilkan oleh software lain. Misalnya saja kita hendak menggambar wajah orang. Anda tinggal mengaktifkan INSET dengan memanggilnya melalui DOS PROMPT. Masuklah ke suatu software untuk menggambar, misalnya DELUXE PAINT. Setelah Anda melakukan penggambaran tekan tombol HOT-KEY untuk mengaktifkan INSET. Maka secara otomatis akan terlihat menu INSET pada layar monitor. Di sini Anda dapat melakukan penyimpanan gambar apa saja yang ada di layar monitor ke dalam file. Tentu saja file ini nantinya dapat Anda buka pada saat Anda penyuntingan. Gambar itu tidak dibuat pada software WORDSTAR sendiri. Sekarang kita akan mencoba membuat program seperti INSET menggunakan TURBO PASCAL.

Listing Program

```
{-------------------------------------------------------------}
{ File        :  Cap2.Pas                                     }
{ Compiler    :  Turbo Pascal 7.00                            }
{ Fungsi      :  Untuk melakukan proses capture gambar.       }
{-------------------------------------------------------------}

{$m 5000,0,0}
```

```
Uses Crt,Dos;

Var Reg: Registers;
    Sudah: Char;
    S,A: Integer;
    R,G,B: Byte;
    C : File;
    Sd: Longint;
    SS: String;
    DirC: String;
    Pal: Text;

Procedure Get;
Interrupt;
  Begin
    If sudah='h' then
      Begin
        s:=s+1;
        Str(s,ss);
        Assign(Pal,DirC+'\'+ss+'.Pal');
        Rewrite(Pal);
        For A:=0 to 255 do
          Begin
            Reg.Ah:=$10;
            Reg.Al:=$15;
            Reg.Bx:=A;
            Intr($10,Reg);
            R:=Reg.Dh;G:=Reg.Ch;B:=Reg.Cl;
            Write(Pal,Chr(R),Chr(G),Chr(B));
          End;
        Close(Pal);
        Assign(C,DirC+'\'+ss+'.Pic');
        Rewrite(C,1);
        BlockWrite(C,Mem[$A000:0],64000);
        Close(C);
      End;
    sudah:='h';
  End;

Begin
    S:=0;
    DirC:=ParamStr(1);
    Sudah:='b';
    Setintvec($5,@get);
    Intr($5,reg);
    KEEP(0);
End.
```

Penjelasan Listing Program

Ini adalah program Capture kita yang akan kita manfaatkan untuk mengambil gambar dari software-software graphic. Untuk itu disarankan agar Anda mempergunakan DELUXE PAINT software. Karena software ini dirancang khusus oleh Electronic Arts untuk pembuatan game dan animasi. Pada software ini Anda dapat merancang animasi per-layar, perubahan palette, pembesaran dan pengecilan gambar, rotasi dengan sudut bebas dan masih banyak lagi fasilitas yang lainnya.

Program Capture ini adalah termasuk program Resident. Yaitu program yang akan bercokol di memori (interrupt) pada saat Anda memanggilnya. Dan apabila Anda menekan tombol tertentu, dia akan muncul dan melaksanakan proses yang telah diinstruksikan.

Cobalah jalankan program ini. Anda harus menjalankannya dari DOS PROMPT secara langsung (Jangan pada editor PASCAL). Berarti Anda harus mengcompilenya terlebih dahulu menjadi program EXE.

Sintaks : CAP2 [PATH]
Contoh : CAP2 C:\PICTURE
Arti : File gambar akan disimpan pada direktori C:\PICTURE.

Setelah Anda jalankan, layar akan diset ke mode graphic 13h. Panggilah software graphic yang akan Anda pergunakan (Software graphic yang Anda pakai harus berada pada mode 320 x 200 256 warna atau mode 13h). Buatlah gambar sembarang kemudian tekan tombol 'SHIFT' + 'PRINT SCREEN'. Tunggulah beberapa saat sampai lampu hard disk berhenti berkedip. Sekarang gambar telah terekam dalam file. Adapun nama file yang dibuat adalah '1.PIC' untuk file gambar dan '1.PAL' untuk file palettenya. Apabila Anda melakukan perekaman gambar lagi dengan 'SHIFT'+ 'PRINT SCREEN', file baru akan disimpan dengan nama '2.PIC ' dan '2.Pal'. Demikian seterusnya selama Anda tidak mereset komputer Anda (Ini akan menyebabkan penyimpanan gambar mulai dari '1.PIC' dan '1.PAL').

Langkah terakhir tinggal mempergunakan file gambar dan file palette tersebut pada program Anda.

PENJELASAN PROGRAM

Karena program ini adalah program resident. Kita harus menyediakan memori tempat program tersebut disimpan. Kita asumsikan saja jumlah memori yang akan dipakai ialah 5000 bytes).

```
{$m 5000,0,0}
```

Selanjutnya kita masuk ke main program.

Set nama gambar dan palette awal dengan 0. Variabel S menunjukkan jumlah gambar yang telah direkam ke file.

```
Begin
    S:=0;
```

Ambil parameter pertama sebagai direktori tempat penyimpanan file gambar dan file palette.

```
    DirC:=ParamStr(1);
```

Proses perekaman belum dilakukan.

```
    Sudah:='b';
```

Arahkan Interrupt vektor 5h untuk tombol SHIFT + PRINT SCREEN ke alamat procedure GET. Biasanya pada saat tombol SHIFT + PRINT SCREEN ditekan komputer akan menjalankan program untuk mencetak layar ke printer. Alamat program pencetak ini kita isi dengan alamat procedure kita (alamat procedure GET) sehingga pada waktu tombol SHIFT + PRINT SCREEN ditekan komputer akan salah alamat, bukannya menjalankan program pencetak layar ke printer, tetapi menjalankan procedure GET.

```
    Setintvec($5,@get);
```

Sekarang kita panggil Interrupt nomor 5h yaitu mencetak layar yang sudah berganti alamat menjadi procedure GET. Tapi di sini masih belum dilakukan perekaman gambar karena nilai sudah = 'b'. Lihat pada procedure Get. Di sana kita beri testing apakah sudah ='h'. Jadi procedure tersebut hanya dilewati sepintas. Kemudian nilai sudah diisi

dengan 'h'. Procedure GET telah siap mengambil gambar apabila tombol SHIFT + PRINT SCREEN ditekan lagi.

```
    Intr($5,reg);
```

Masukkan procedure GET ke dalam Interrupt 5h.

```
    KEEP(0);
End.
```

Sekarang lihatlah procedure GET.

```
Procedure Get;
```

Tipe procedure ini adalah interrupt.

```
Interrupt;
  Begin
```

Apakah variabel SUDAH berharga 'h'.

```
    If sudah='h' then
      Begin
```

Tambahkan jumlah gambar yang telah direkam.

```
        s:=s+1;
```

Konversi jumlah gambar yang telah direkam ke STRING supaya bisa dipergunakan sebagai nama file.

```
        Str(s,ss);
```

Buka file palette dan rekam warna palette mulai nomor 0 hingga 255 ke dalam file.

```
        Assign(Pal,DirC+'\'+ss+'.Pal');
        Rewrite(Pal);
        For A:=0 to 255 do
          Begin
            Reg.Ah:=$10;
            Reg.Al:=$15;
            Reg.Bx:=A;
```

```
            Intr($10,Reg);
            R:=Reg.Dh;G:=Reg.Ch;B:=Reg.Cl;
            Write(Pal,Chr(R),Chr(G),Chr(B));
          End;
        Close(Pal);
```

Buka file gambar dan rekam data gambar ke dalam file.

```
        Assign(C,DirC+'\'+ss+'.Pic');
        Rewrite(C,1);
        BlockWrite(C,Mem[$A000:0],64000);
        Close(C);
      End;
```

Variabel SUDAH berharga 'h' yang berarti procedure GET siap untuk mengambil gambar yang ada di layar apabila dilakukan penekanan tombol SHIFT + PRINT SCREEN.

```
    sudah:='h';
  End;
```

Sekarang baru kita mulai mencoba membuat animasi dengan gambar yang benar-benar nyata. Coba jalankan program di bawah ini. Tapi ingat Anda harus menjalankan program ini pada direktori dimana disket "PEMROGRAMAN ANIMASI DAN GAME PROFESIONAL" diinstal, karena program ini membutuhkan file gambar dan file palette yang berada pada direktori tersebut.

Listing Program

```
{-------------------------------------------------------------}
{File      : Real1.Pas                                        }
{Compiler  : Turbo Pascal 7.00                                }
{Fungsi    : Untuk menggerakkan pesawat dengan teknik animasi tiga layar.  }
{-------------------------------------------------------------}
Uses Crt,Dos,VgaCad;
Const
    Up    = 1;
    Down  = 2;
    Left  = 3;
    Right= 4;

Var Back,Process: Pointer;
    Ship       : Pointer;
    Size       : Word;
```

```
    A,B         : Integer;
    Dir         : Byte;
    Ch          : Char;
    Luar        : Boolean;

Procedure Initialize;

  Begin
    Define_Page(Back);
    Define_Page(Process);
    A:=160;B:=100;
    Luar:=False;
  End;

Procedure Generate_Picture;

  Begin
    Load_Picture('Back2.Pic',Back^);
    Load_Picture('Ship.Pic',Screen^);
    GetMem(Ship,ImageSize(1,1,22,20));
    GetImage(Screen^,1,1,22,20,Ship^);
  End;

Procedure Main;

  Begin
    Repeat
      Repeat
        MoveTo(Back^,Process^);
        Case Dir of
          Up:If B>0 then Dec(B,1);
          Down:If B<175 then Inc(B,1);
          Left:If A>0 then Dec(A,1);
          Right:If A<295 then Inc(A,1);
        End;
        PutImage(Process^,A,B,Ship^);
        MoveTo(Process^,Screen^);
      Until Keypressed;
      Ch:=Readkey;
      Case Ch of
        #80:Dir:=Down;
        #72:Dir:=Up;
        #75:Dir:=Left;
        #77:Dir:=Right;
        #27:Luar:=True;
        ' ':Dir:=0;
      End;
    Until Luar;
  End;
```

```
Begin
  OpenVga256;
  Install_Palette('Back2.Pal');
  Initialize;
  Generate_Picture;
  Main;
  CloseVga256;
End.
```

Penjelasan Listing Program

Bagaimana? Puaskah Anda? Tanpa sadar Anda telah mulai menjajaki level pembuatan game bertipe RAPTOR. Memang untuk mempelajari teknik-teknik di atas cukup sulit. Tetapi janganlah putus asa di tengah jalan, sebentar lagi kita akan menggabungkan semua teknik yang telah kita pelajari untuk membuat game yang baik. Dan pastilah lebih menyenangkan apabila Anda telah menguasainya.

PENJELASAN PROGRAM

Sebagian besar program di atas telah Anda kuasai karena hampir sama dengan program-program terdahulu. Tujuan kita mempelajari tipe-tipe program yang tidak banyak berubah adalah untuk mempermudah Anda mempelajari teknik baru. Untuk itu Anda harus mengerti bagian-bagian awal dari buku ini supaya Anda memiliki dasar yang benar dan kuat.

Procedure yang perlu kita pelajari ialah GENERATE_PICTURE karena di sinilah letak perbedaan program ini dengan program sebelumnya (3SCR1.PAS dan 3SCR3.PAS).

```
Procedure Generate_Picture;
  Begin
```

Load gambar background ke background page. Dengan ini maka kita tidak perlu melakukan penggambaran background dengan bangun-bangun geometri. Apalagi kalau gambar background yang diinginkan seperti pada program ini (Hampir tidak mungkin untuk dibuat dengan menggunakan fungsi-fungsi geometri).

```
    Load_Picture('Back2.Pic',Back^);
```

Load gambar pesawat ke visual page.

```
    Load_Picture('Ship.Pic',Screen^);
```

Sediakan memori sebesar IMAGESIZE pada alamat pointer Ship.

```
    GetMem(Ship,ImageSize(1,1,22,20));
```

Ambil gambar dengan GETIMAGE dari visual page ke alamat pointer Ship.

```
    GetImage(Screen^,1,1,22,20,Ship^);
  End;
```

Anda tentu melihat sekilas pada saat dilakukan load gambar pesawat ke visual page dalam rangka mengadakan GETIMAGE. Hal ini akan memperburuk tampilan. Kita tidak ingin gambar tersebut keluar pada visual page. Untuk itu Anda dapat mengganti procedure di atas dengan procedure ini.

```
Procedure Generate_Picture;
  Begin
    Load_Picture('Back2.Pic',Back^);
    Load_Picture('Ship.Pic',Process^);
    GetMem(Ship,ImageSize(1,1,22,20));
    GetImage(Process^,1,1,22,20,Ship^);
  End;
```

Pada procedure ini dilakukan load gambar pesawat ke process page sehingga gambar tersebut tidak kita lihat. Kemudian dilakukan GETIMAGE dari Process page pula. Dengan trik ini maka tampilan animasi kita tidak akan terlihat amatiran.

Untuk sementara kita tinggalkan dahulu mengenai pengolahan gambar. Mungkin ada baiknya kalau Anda mencoba-coba dahulu untuk membuat animasi dengan bantuan software lain seperti yang telah kita lakukan pada program di atas.

APLIKASI ANIMASI PALETTE

Pada bab sebelumnya kita telah mempelajari apa yang dimaksud dengan PALETTE. Apa kegunaannya, bagaimana cara pengaturannya dan sebagainya.

Di samping menyediakan fasilitas-fasilitas pembuatan gambar, unit VGACAD.TPU juga dilengkapi dengan procedure-procedure pengolahan palette yang sangat efektif. Dengan ini Anda dapat mempermainkan palette seperti pengaturan brightness (tingkat terangnya gambar) dan contrast (tingkat ketajaman gambar) seperti fasilitas PHOTO STYLER. Anda juga bisa menciptakan effek silau, perubahan warna secara perlahan-lahan dan lain sebagainya.

Pada Unit VGACAD.TPU ini telah disediakan variabel bertipe Palette_Addr yang merupakan pointer yang mengarah pada penyimpanan palette (PALETTE_TYPE). Active_Pal, Defult_Pal dan User_Pal adalah variabel bertipe Palette_Addr. Pada ketiga alamat pointer ini disimpan data intensitas palette secara lengkap (warna 0 hingga 255). Tetapi ingat baik-baik , susunan intensitas palette yang sedang aktif tidak ditentukan oleh ketiga variabel tersebut. Seperti virtual palette (sama halnya seperti animasi tiga layar dengan virtual pagenya, virtual page bukan merupakan page yang kita lihat tetapi kita dapat menduplikasi virtual page tersebut ke visual page sehingga dapat kita lihat). Default_Pal berisi susunan palette pada saat kita melakukan instalasi palette dengan procedure INSTALL_PALETTE. Tujuannya apabila kita telah melakukan perubahan intensitas palette dan hendak kembali ke palette awal, tinggal memanggil procedure RESTORE_PALETTE yang akan mengembalikan susunan palette ke palette yang disimpan di Default_Pal. Selama Anda tidak melakukan perubahan manual (Dengan SetPalette atau SetAllPalette untuk mengubah susunan palette dalam Active_Pal), Active_Pal menyimpan susunan palette yang sedang aktif di layar monitor. Apabila telah terjadi perubahan pada Active_Pal, Anda dapat memanggil procedure REFRESH_PALETTE untuk mengaktifkan susunan palette dalam Active_Pal ke layar monitor. Dengan adanya User_Pal, Anda dapat merencanakan susunan palette yang Anda inginkan.

FASILITAS MANIPULASI PALETTE PADA VGACAD.TPU

PROCEDURE SET VISUAL PALETTE

Sintaks : SetVisualPalette(No,R,G,B);
Variabel : No = Nomor Palette yang hendak diubah.
R,G,B = Intensitas warna Merah, Hijau dan Biru yang hendak dimasukkan. Nilai maksimum masing-masing ialah 63.
Fungsi : Untuk mengubah warna palette langsung ke layar monitor.

Tentunya Anda sudah mengetahui sistem perubahan warna palette dengan port di bawah ini (Telah dibahas pada bab sebelumnya).

```
Procedure SetVisualPalette (No,R,G,B:Byte);
  Begin
    Port[$3c8]:=No;          { Port nomor Palette      }
    Port[$3c9]:=R;           { Isi Port Merah          }
    Port[$3c9]:=G;           { Isi Port Hijau          }
    Port[$3c9]:=B;           { Isi Port Biru           }
  End;
```

PROCEDURE GET VISUAL PALETTE

Sintaks : GetVisualPalette (No,R,G,B);
Variabel : No = Nomor palette yang akan dibaca.
R,G,B = Variabel output yang akan mengeluarkan intensitas warna merah, hijau dan biru.
Fungsi : Untuk membaca intensitas palette pada palette nomor NO.

Sistem pembacaan palette dengan Port ini juga telah kita bahas pada bab tentang aplikasi palette pada monitor VGA.

```
Procedure GetVisualPalette (No:Byte; Var R,G,B:Byte);
  Begin
    Port[$3c8]:=No;          { Port nomor Palette      }
    R:=Port[$3c9];           { Baca Port Merah         }
    G:=Port[$3c9];           { Baca Port Hijau         }
    B:=Port[$3c9];           { Baca Port Biru          }
  End;
```

PROCEDURE SET PALETTE

Sintaks : SetPalette (Palette_Buff,No,R,G,B);

Variabel : Palette_Buff = Virtual Palette yang hendak diubah.
No = Nomor palette yang hendak diubah.
R,G,B = Intensitas merah, hijau dan biru dari palette yang hendak dimasukkan.

Fungsi : Untuk mengubah intensitas palette dalam virtual Palette. Perubahan yang dilakukan di sini tidak mengubah intensitas palette yang sedang aktif di layar monitor.

```
Procedure SetPalette    (Arr_Pal:Palette_Addr; No,Rr,Gg,Bb:Byte);
  Begin
    With Arr_Pal^[No] do
      Begin
        R:=Rr;
        G:=Gg;
        B:=Bb;
      End;
  End;
```

PROCEDURE GET PALETTE

Sintaks : GetPalette (Palette_Buff,No,R,G,B);

Variabel : Palette_Buff = Virtual Palette yang hendak dibaca.
No = Nomor Palette yang hendak dibaca.
R,G,B = Variabel output yang mengeluarkan intensitas merah, hijau dan biru pada nomor palette NO.

Fungsi : Untuk membaca intensitas palette pada virtual palette nomor palette = NO.

```
Procedure GetPalette        (Arr_Pal:Palette_Addr; No:Byte; Var
Rr,Gg,Bb:Byte);
  Begin
    With Arr_Pal^[No] do
      Begin
        Rr:=R;
        Gg:=G;
        Bb:=B;
      End;
  End;
```

PROCEDURE LOAD PALETTE

Sintaks : Load_Palette(Palette_Buff,Nama_File);
Variabel : Palette_Buff = Virtual Palette tempat dimana palette akan diisikan dari file.
Nama_File = Nama file dimana susunan palette akan dibaca.
Fungsi : Untuk membaca susunan palette dari file ke virtual palette.

```
Procedure Load_Palette(Pal_Buff:Pointer; File_Name:String);
Var X      : Byte;
    R,G,B  : Char;
    RR,GG,BB: Byte;
  Begin
    Assign(File_Pal,File_Name);
    Reset(File_Pal,1);
```

Baca File Palette ke alamat pointer BUFF sejumlah 768 bytes (256 x 3). Rumus 256 x 3 didapat dari 256 warna palette x (R + G + B) di mana R, G dan B masing-masingnya ialah variabel 1 byte.

```
    BlockRead(File_Pal,Pal_Buff^,768);
    Close(File_Pal);
  End;
```

PROCEDURE SAVE PALETTE

Sintaks : Save_Palette (Palette_Buff,Nama_File);
Variabel : Palette_Buff = Virtual Palette yang akan disimpan.
Nama_File = Nama file tempat virtual Palette akan disimpan.
Fungsi : Untuk menyimpan virtual palette ke dalam file.

```
Procedure Save_Palette (Pal_Buff:Pointer; File_Name:String);
Var X      : Byte;
    R,G,B  : Byte;
  Begin
    Assign(File_Pal,File_Name);
    Rewrite(File_Pal,1);
    BlockWrite(File_Pal,Pal_Buff^,768);
    Close(File_Pal);
  End;
```

PROCEDURE REFRESH PALETTE

Sintaks : RefreshPalette;
Fungsi : Untuk mengaktifkan Active_Pal ke layar monitor. Sehingga warna palette yang muncul di monitor = Active_Pal.

```
Procedure Refresh_Palette;
  Begin
    For A:=0 to 255 do
      With Active_Pal^[A] do
        Begin
```

Aktifkan palette nomor A dari Active_Pal ke layar monitor.

```
          SetVisualPalette(A,R,G,B);
        End;
  End;
```

PROCEDURE ACTIVE PALETTE

Sintaks : ActivePalette(Palette_Buff);
Variabel : Palette_Buff = Virtual page yang akan diaktifkan ke layar monitor.
Fungsi : Untuk mengaktifkan virtual palette ke layar monitor sekalian dengan mengisi Active_Pal dengan virtual palette tersebut.

```
Procedure Active_Palette(Ac_Pal:Palette_Addr);
  Begin
```

Lakukan penduplikatan isi virtual palette ke Active_Pal secara Word sehingga jumlah Word yang ditransfer = 768/2 = 384.

```
    MoveW(Ac_Pal^,Active_Pal^,384);
```

Aktifkan Active_Pal ke layar monitor.

```
    Refresh_Palette;
  End;
```

PROCEDURE RESTORE PALETTE

Sintaks : Restore_Palette;
Fungsi : Untuk mengembalikan Active_Pal ke Default_Pal. Di sini palette di layar masih belum mengalami perubahan. Untuk itu Anda perlu melakukan pemanggilan REFRESH_PALETTE.

```
Procedure Restore_Palette;
  Begin
    MoveW(Default_Pal^,Active_Pal^,384);
  End;
```

PROCEDURE INSTALL PALETTE

Sintaks : Install_Palette(Nama_File);
Variabel : Nama_File = Nama file palette yang akan dibaca sebagai Default_Palette.
Fungsi : Untuk menciptakan Default_Pal, User_Pal dan Active_Pal. Juga untuk mengisi Default_Pal dengan nama file yang sesuai dengan parameter yang dikirimkan.

Install_Palette ini tidak boleh dilakukan lebih dari satu kali karena hal ini akan menimbulkan pengulangan GETMEM pada alamat pointer yang sama.

```
Procedure Install_Palette(Name:String);
  Begin
```

Pesan memori untuk Default_Pal,User_Pal dan Active_Pal.

```
    GetMem(Default_Pal,768);
    GetMem(User_Pal,768);
    GetMem(Active_Pal,768);
```

Baca Default_Palette dari file.

```
    Load_Palette(Default_Pal,Name);
```

Aktifkan susunan palette pada layar monitor dengan Default_Pal. Juga isi Active_Pal dengan Default_Pal (Hal ini dilakukan di dalam procedure ACTIVE_PALETTE).

```
    Active_Palette(Default_Pal);
  End;
```

PROCEDURE SET ALL PALETTE

Sintaks : Set_All_Palette (Palette_Buff,R,G,B);
Variabel : Palette_Buff = Virtual Palette yang akan diset.
R,G,B = Intensitas merah, hijau dan biru yang hendak diisikan pada semua nomor palette.
Fungsi : Untuk mengisi semua (0 sampai 255) intensitas palette pada Virtual Palette dengan parameter yang diberikan.

```
Procedure Set_All_Palette   (Pal_Buff:Palette_Addr; R,G,B: Byte);
  Begin
    For A:=0 to 255 do
      SetPalette(Pal_Buff,A,R,G,B);
  End;
```

PROCEDURE ACTIVE TO PALETTE

Sintaks : Active_To_Palette(Palette_Buff,Delay);
Variabel : Palette_Buff = Virtual Palette yang dituju.
Delay = Jumlah selang waktu untuk Delay.
Fungsi : Untuk mengubah palette yang sedang aktif ke palette tujuan (Palette_Buff) secara perlahan-lahan.

Sistem pembuatannya sangat mudah. Kita cukup menggeser seluruh intensitas warna palette yang sedang aktif (Active_Pal) mendekati Palette tujuan (Palette_Dest). Misalnya saja palette tujuan dengan nomor palette 5 memiliki intensitas warna merah = 60. Sedangkan palette yang sedang aktif (Active_Pal), juga nomor palette 5 memiliki intensitas warna merah = 30. Kemudian dilakukan penambahan intensitas warna palette ini dengan 1 karena palette tujuan memiliki intensitas lebih besar. Lain halnya kalau palette tujuan pada nomor palette 5 memiliki intensitas warna merah = 10. Tentu saja pada Active_Pal nomor palette 5 tidak dilakukan penambahan melainkan dilakukan pengurangan dengan 1. Perubahan ini dilakukan untuk setiap intensitas warna palette pada Active_Pal, baik merah, hijau maupun biru (untuk palette nomor 0 hingga 255). Setiap dilakukan perubahan setiap intensitas palette, diadakan pemanggilan REFRESH_PALETTE untuk mengaktifkan palette tersebut ke layar monitor sehingga perubahan yang terjadi dapat kita lihat perlahan-lahan. Proses ini diulang terus hingga semua intensitas palette dalam Active_Pal sama dengan palette tujuan.

```
Procedure Active_To_Palette(Pal_Dest:Palette_Addr; Del:Byte);
Var Done   : Boolean;
    A      : Integer;
  Begin
    Repeat
```

Diasumsikan semua Active_Pal telah berubah menjadi Palette_Dest.

```
      Done:=True;
```

Tunggu beberapa saat untuk memperlambat proses perubahan palette. Karena proses ini sangat cepat, sehingga hampir-hampir kita tidak dapat melihatnya tanpa diberi DELAY.

```
      Delay(Del);
```

Ubah seluruh intensitas warna palette pada Active_Pal.

```
      For A:=0 to 255 do
        With Pal_Dest^[A] do
          Begin
            If Active_Pal^[A].R<R then Inc(Active_Pal^[A].R);
            If Active_Pal^[A].R>R then Dec(Active_Pal^[A].R);
            If Active_Pal^[A].G<G then Inc(Active_Pal^[A].G);
            If Active_Pal^[A].G>G then Dec(Active_Pal^[A].G);
            If Active_Pal^[A].B<B then Inc(Active_Pal^[A].B);
            If Active_Pal^[A].B>B then Dec(Active_Pal^[A].B);
```

Apabila ada satu saja dari intensitas merah, hijau dan biru dari Active_Pal yang tidak sama dengan intensitas palette tujuan, maka DONE diberi nilai False sehingga loop tidak dihentikan.

```
            If Active_Pal^[A].R<>R then Done:=False;
            If Active_Pal^[A].G<>G then Done:=False;
            If Active_Pal^[A].B<>B then Done:=False;
          End;
```

Aktifkan palette Active_Pal yang telah diubah ke layar monitor.

```
      Refresh_Palette;
```

Sampai semua intensitas palette dalam Active_Pal telah berubah menjadi palette tujuan.

```
  Until Done;
End;
```

FUNCTION METAMORPH PALETTE

Sintaks : Done:=Metamorph_Palette(Palette_Buff,Palette_Dest,Rate);
Variabel : Palette_Buff = Virtual Palette yang hendak diubah.
Palette_Dest = Virtual Palette yang dituju.
Rate = Perubahan intensitas yang hendak dilakukan.
Fungsi : Untuk mengubah virtual palette (Palette_Buff) ke virtual palette tujuan (Palette_Dest) sejumlah Rate.

Sebenarnya Metamorph Palette ini sama dengan Active_To_Palette tetapi di sini cuma dilakukan perubahan intensitas warna sejumlah RATE. Perubahan ini juga tidak terbatas hanya untuk mengubah Active_Pal, melainkan virtual palette yang Anda kirim lewat parameter. Setelah itu program keluar dari function. Jika Anda ingin melakukan perubahan hingga Palette_Buff berubah total menjadi Palette_Dest, Anda harus melakukan looping untuk memanggil function ini berkali-kali hingga dia mengeluarkan nilai TRUE yang berarti semua palette telah berubah total. Dengan cara ini maka kita dapat melakukan perubahan warna palette sambil melakukan kegiatan lainnya secara bersamaan (Misalnya diiringi dengan menggerakkan sprite).

Sistem pembuatan procedure ini hampir sama dengan pembuatan procedure Active_To_Palette. Di sini perubahan intensitas virtual palette hanya dilakukan sekali tanpa peduli apakah semua intensitas virtual palette tersebut sudah sama dengan Palette tujuan.

```
Function  Metamorph_Palette(Pal_Buff,  Pal_Dest:  Palette_Addr;
Rate:Byte): Boolean;
  Begin
```

Diasumsikan perubahan intensitas warna palette telah selesai.

```
    Metamorph_Palette:=True;
```

Ubah seluruh intensitas warna palette mulai 0 hingga 255.

```
      For A:=0 to 255 do
        With Pal_Dest^[A] do
          Begin
```

```
        If Pal_Buff^[A].R<R then
          Begin
               If Pal_Buff^[A].R+Rate<R then Inc(Pal_Buff^[A].R,Rate)
               Else Pal_Buff^[A].R:=R;
          End;
        If Pal_Buff^[A].R>R then
          Begin
               If Pal_Buff^[A].R-Rate>R then Dec(Pal_Buff^[A].R,Rate)
               Else Pal_Buff^[A].R:=R;
          End;
        If Pal_Buff^[A].G<G then
          Begin
               If Pal_Buff^[A].G+Rate<G then Inc(Pal_Buff^[A].G,Rate)
               Else Pal_Buff^[A].G:=G;
          End;
        If Pal_Buff^[A].G>G then
          Begin
               If Pal_Buff^[A].G-Rate>G then Dec(Pal_Buff^[A].G,Rate)
               Else Pal_Buff^[A].G:=G;
          End;
        If Pal_Buff^[A].B<B then
          Begin
               If Pal_Buff^[A].B+Rate<B then Inc(Pal_Buff^[A].B,Rate)
               Else Pal_Buff^[A].B:=B;
          End;
        If Pal_Buff^[A].B>B then
          Begin
               If Pal_Buff^[A].B-Rate>B then Dec(Pal_Buff^[A].B,Rate)
               Else Pal_Buff^[A].B:=B;
          End;
```

Apabila ada satu saja dari intensitas merah, hijau dan biru dari Pal_Buff yang tidak sama dengan intensitas palette tujuan, maka Metamorph_Palette diberi nilai FALSE sehingga kita akan memperoleh nilai function Metamorph_Palette = FALSE.

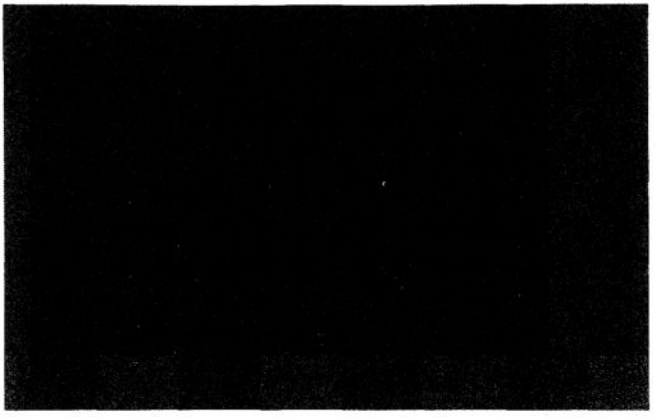
```
        If Pal_Buff^[A].R<>R then Metamorph_Palette:=False;
        If Pal_Buff^[A].G<>G then Metamorph_Palette:=False;
        If Pal_Buff^[A].B<>B then Metamorph_Palette:=False;
      End;
End;
```

PEMANFAATAN FASILITAS PALETTE PADA VGACAD.TPU

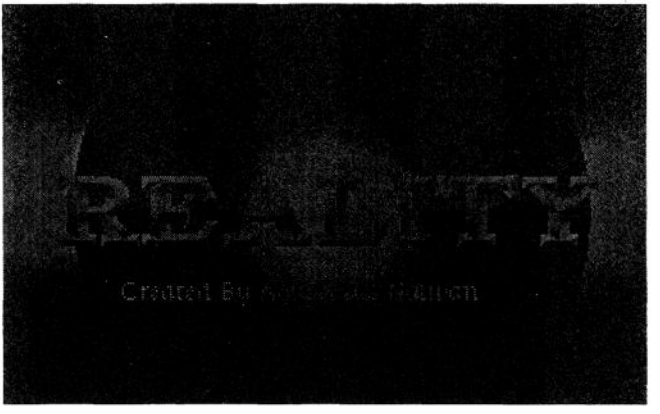
Sekarang kita akan mencoba procedure-procedure di atas untuk pembuatan animasi. Tentu saja animasi yang ditonjolkan di sini adalah permainan palette. Coba lihat program berikut ini.

Listing Program

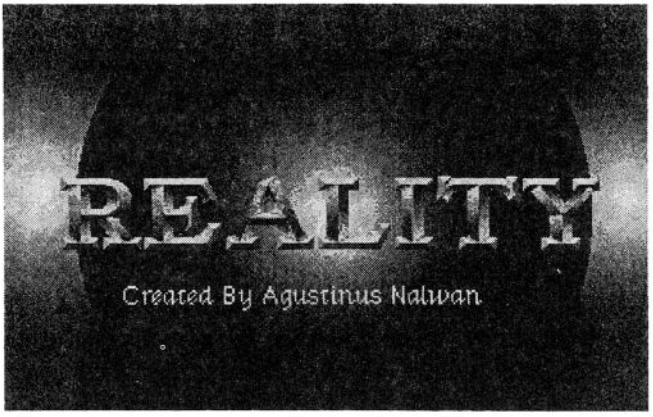
```
{-------------------------------------------------------------}
{ File     :  Pal1.Pas                                        }
{ Compiler :  Turbo Pascal 7.00                               }
{ Fungsi   :  Untuk melakukan animasi palette sederhana       }
{-------------------------------------------------------------}

Uses Crt,Dos,VgaCad;

Var Ch:Char;

Begin
  OpenVga256;
  Install_Palette('PALETTE.PAL');
  Set_All_Palette(Active_Pal,0,0,0);
  Refresh_Palette;
  Load_Picture('PALETTE.PIC',Screen^);
  Active_To_Palette(Default_Pal,50);
  Set_All_Palette(User_Pal,0,0,0);
  Ch:=Readkey;
  Active_To_Palette(User_Pal,50);
  CloseVga256;
End.
```



Gambar 3.20 Intensitas Palette digeser perlahan-lahan dari hitam total menjadi intensitas yang dituju.



Gambar 3.21 Intensitas palette sudah hampir mendekati intensitas palette yang dituju



Gambar 3.22 Intensitas palette telah mencapai intensitas palette tujuan. Pada saat ini procedure ACTIVE_TO_PALETTE telah selesai melakukan proses.

Untuk menjalankan program ini diperlukan file PALETTE.PAL dan PALETTE.PIC pada directory yang sama yang berisi data gambar dan data palette yang akan dianimasikan. Pertama-tama akan muncul sebuah gambar di layar monitor mulai gelap hingga terang (seakan-akan dari lampu mati, dinaikkan intensitas terangnya secara perlahan-lahan). Kemudian coba tekan ENTER. Gambar akan mulai redup sedikit demi sedikit hingga akhirnya gelap total. Animasi palette ini akan memegang peranan penting dalam pembuatan game.

PENJELASAN PROGRAM

```
Begin
  OpenVga256;
```

Install Default Palette dengan palette dari file PALETTE.PAL.

```
  Install_Palette('PALETTE.PAL');
```

Isi semua intensitas Active_Pal dengan 0 (hitam).

```
  Set_All_Palette(Active_Pal,0,0,0);
```

Aktifkan Active_Pal ke layar monitor sehingga seluruh intensitas palette pada layar monitor berisi warna hitam.

```
Refresh_Palette;
```

Load gambar ke layar monitor. Gambar belum bisa kita lihat karena intensitas palette pada layar monitor semuanya hitam.

```
Load_Picture('PALETTE.PIC',Screen^);
```

Ubah Active_Pal secara perlahan-lahan ke arah Default_Pal. Seluruh intensitas warna pada Active_Pal akan mulai ditambah mulai 0 hingga menjadi seperti Default_Pal. 50 adalah Delay untuk memperlambat proses penerangan palette.

```
Active_To_Palette(Default_Pal,50);
```

Isi User_Pal yang merupakan palette tujuan dengan warna hitam.

```
Set_All_Palette(User_Pal,0,0,0);
```

Tunggu penekanan tombol.

```
Ch:=Readkey;
```

Ubah Active_Pal secara perlahan-lahan ke arah User_Pal sehingga seluruh intensitas warna pada Active_Pal akan dikurangi perlahan-lahan sehingga menjadi warna hitam (User_Pal).

```
  Active_To_Palette(User_Pal,50);
  CloseVga256;
End.
```

Sekarang kita akan mencoba melangkah lebih jauh lagi. Sekarang kita akan mencoba menggunakan METAMORPH_PALETTE. Perhatikan bedanya. Di sini Anda' harus memanggilnya berulang-ulang hingga function tersebut mengeluarkan nilai TRUE yang berarti semua intensitas palette telah diubah. Dengan cara ini, Anda dapat menyisipkan program lain dalam looping (misalnya disertai dengan penggerakkan sprite).

Listing Program

```
{-------------------------------------------------------------}
{ File     :  Pal2.Pas                                        }
{ Compiler:  Turbo Pascal 7.00                                }
{ Fungsi   : Untuk melakukan animasi palette dengan fungsi metamorph palette.}
{-------------------------------------------------------------}
Uses Crt,Dos,VgaCad;
Var Done: Boolean;

Begin
  OpenVga256;
  Install_Palette('PALETTE.PAL');
  Load_Picture('PALETTE.PIC',Screen^);
  Set_All_Palette(User_Pal,60,0,0);
  Repeat
    Repeat
      Done:=Metamorph_Palette(Active_Pal,User_Pal,1);
      Refresh_Palette;
      Delay(20);
    Until Done;
    Repeat
      Done:=Metamorph_Palette(Active_Pal,Default_Pal,1);
      Refresh_Palette;
      Delay(20);
    Until Done;
    Set_All_Palette(User_Pal,60,0,60);
    Repeat
      Done:=Metamorph_Palette(Active_Pal,User_Pal,1);
      Refresh_Palette;
      Delay(20);
    Until Done;
   Repeat
      Done:=Metamorph_Palette(Active_Pal,Default_Pal,1);
      Refresh_Palette;
      Delay(20);
    Until Done;
  Until Keypressed;
  Set_All_Palette(User_Pal,60,0,0);
  Active_To_Palette(User_Pal,10);
  CloseVga256;
End.
```

Penjelasan Listing Program

Program ini pada dasarnya sama dengan program PALETTE1. Di sini kita mempergunakan function METAMORPH_PALETTE. Gambar yang

dikeluarkan di layar akan berubah secara perlahan menjadi merah kemudian kembali normal, kemudian menjadi ungu dan kembali normal lagi. Demikianlah seterusnya.

```
Begin
  OpenVga256;
```

Install Default Palette dari file PALETTE.PAL.

```
  Install_Palette('PALETTE.PAL');
```

Load gambar PALETTE.PIC ke layar monitor.

```
  Load_Picture('PALETTE.PIC',Screen^);
```

Palette tujuan adalah merah.

```
  Set_All_Palette(User_Pal,60,0,0);
  Repeat
```

Ulangi proses di bawah ini.

```
    Repeat
```

Panggil Procedure METAMORPH_PALETTE untuk mengubah Active_Pal ke arah User_Pal.

```
      Done:=Metamorph_Palette(Active_Pal,User_Pal,1);
```

Aktifkan Active_Pal ke layar monitor. Karena pada Active_Pal telah diadakan perubahan ke arah intensitas merah, maka palette pada layar monitor akan berubah menjadi semakin merah.

```
      Refresh_Palette;
```

Beri selang waktu untuk memperlambat proses.

```
      Delay(20);
```

Sampai semua Active_Pal telah menjadi User_Pal (Merah).

```
    Until Done;
```

Pada saat ini seluruh intensitas palette di layar monitor dan di Active_Pal adalah merah (Intensitas merah = 60 sedangkan intensitas hijau dan biru = 0). Ulangi proses di bawah ini.

```
Repeat
```

Ubah Active_Pal ke arah Default_Pal sehingga gambar akan kembali normal secara perlahan-lahan.

```
Done:=Metamorph_Palette(Active_Pal,Default_Pal,1);
```

Aktifkan Active_Pal ke layar monitor.

```
Refresh_Palette;
```

Beri selang waktu untuk memperlambat proses.

```
Delay(20);
```

Hingga seluruh Active_Pal telah menjadi Default_Pal.

```
Until Done;
```

Proses selanjutnya sama saja dengan proses di atas, bedanya di sini intensitas warna diubah menjadi ungu kemudian kembali normal.

```
Set_All_Palette(User_Pal,60,0,60);
Repeat
  Done:=Metamorph_Palette(Active_Pal,User_Pal,1);
  Refresh_Palette;
  Delay(20);
 Until Done;
Repeat
  Done:=Metamorph_Palette(Active_Pal,Default_Pal,1);
  Refresh_Palette;
  Delay(20);
 Until Done;
Until Keypressed;
```

Palette tujuan adalah hitam.

```
Set_All_Palette(User_Pal,60,0,0);
```

Ubah Active_Pal ke User_Pal (Hitam). Ini berguna untuk meredupkan layar sebelum program selesai.

```
  Active_To_Palette(User_Pal,10);
  CloseVga256;
End.
```

Pada televisi dikenal istilah *brightness* yang berarti menambah tingkat terang dari gambar yang dihasilkan. Fasilitas ini juga ada pada layar monitor Anda tentunya. Coba cari di bagian belakang, samping, depan atau bawah dari monitor Anda, pasti ada tombol *brightness*. Apabila tombol ini Anda putar-putar maka monitor Anda akan menjadi redup-terang-redup-terang. Di samping itu, pada beberapa monitor ada juga tombol yang berfungsi untuk meningkatkan intensitas RGB, biasanya ada tiga tombol. Anda dapat mengatur intensitas warna merah, hijau atau biru yang ditampilkan di layar monitor. Sekarang ini kita akan mencoba memanfaatkan palette untuk membuat fasilitas-fasilitas tersebut secara programming.

Listing Program

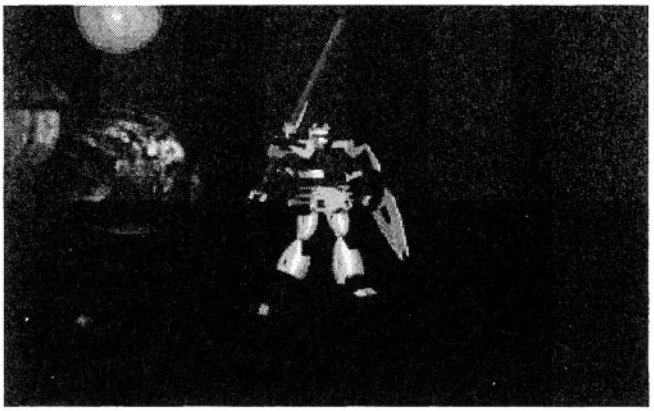
```
{-------------------------------------------------------------}
{ File     : Pal3.Pas                                         }
{ Compiler : Turbo Pascal 7.00                                }
{ Fungsi   : Untuk mengubah intensitas palette seperti televisi.}
{-------------------------------------------------------------}

Uses Crt,Dos,VgaCad;

Var Luar: Boolean;
    Ch: Char;

Procedure Initialize;
  Begin
    Install_Palette('HERO.PAL');
    Set_All_Palette(Active_Pal,0,0,0);
    Refresh_Palette;
    Load_Picture('HERO.PIC',Screen^);
    Active_To_Palette(Default_Pal,40);
    Luar:=False;
  End;
```

```
Procedure Main;
  Begin
    Repeat
      Repeat
      Until Keypressed;
      Ch:=Readkey;
      Case Ch of
        #27:Luar:=True;
        '1':Begin
              Set_All_Palette(User_Pal,63,63,63);
              Metamorph_Palette(Active_Pal,User_Pal,1);
              Refresh_Palette;
            End;
        '2':Begin
              Set_All_Palette(User_Pal,0,0,0);
              Metamorph_Palette(Active_Pal,User_Pal,1);
              Refresh_Palette;
            End;
        'R','r':Begin
              Set_All_Palette(User_Pal,63,0,0);
              Metamorph_Palette(Active_Pal,User_Pal,1);
              Refresh_Palette;
            End;
        'G','g':Begin
              Set_All_Palette(User_Pal,0,63,0);
              Metamorph_Palette(Active_Pal,User_Pal,1);
              Refresh_Palette;
            End;
        'B','b':Begin
              Set_All_Palette(User_Pal,0,0,63);
              Metamorph_Palette(Active_Pal,User_Pal,1);
              Refresh_Palette;
            End;
        ' ':Begin
              Metamorph_Palette(Active_Pal,Default_Pal,1);
              Refresh_Palette;
            End;
      End;
    Until Luar;
    Set_All_Palette(User_Pal,0,0,0);
    Active_To_Palette(User_Pal,30);
  End;

Begin
  OpenVga256;
  Initialize;
  Main;
  CloseVga256;
End.
```

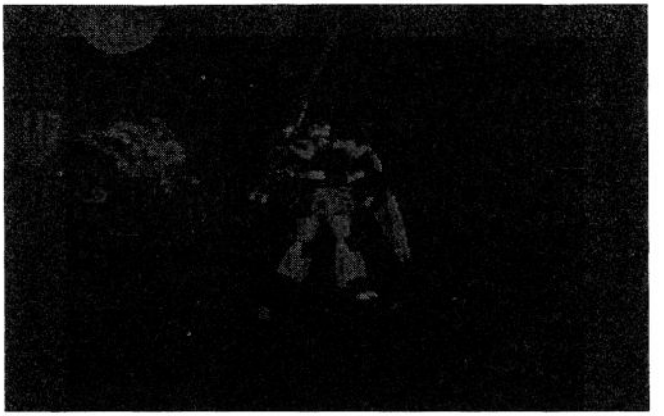
Penjelasan Listing Program

Setelah Anda eksekusi, program ini akan menampilkan robot. Di sini Anda dapat mengatur brightness dan intensitas RGB dari palette.

- Tombol '1' berguna untuk meningkatkan brightness dari gambar.
- Tombol '2' untuk menurunkan brightness dari gambar.
- Tombol 'R' untuk meningkatkan intensitas warna merah.
- Tombol 'G' untuk meningkatkan intensitas warna hijau.
- Tombol 'B' untuk meningkatkan intensitas warna biru.
- Tombol Space untuk mengembalikan gambar ke Default Palette secara perlahan-lahan.
- Tombol ESC untuk keluar dari program.



Gambar 3.23 Intensitas warna merah ditambah sehingga gambar yang dihasilkan menjadi merah menyala.



Gambar 3.24 Intensitas Red, Green maupun Blue dari palette dikurangi sehingga gambar terlihat gelap

Cobalah bermain-main sebentar dengan program ini.

Di sinilah pentingya peranan METAMORPH_PALETTE baru terasa, karena bila tombol ditekan, intensitas palette hanya akan diubah sedikit saja, tidak langsung berubah ke arah palette tujuan seperti bila kita menggunakan procedure ACTIVE_TO_PALETTE.

Procedure yang akan kita bahas adalah procedure MAIN saja, karena sisanya tentu saja Anda sudah mengerti dari pengalaman-pengalaman program yang terdahulu.

```
Procedure Main;
  Begin
    Repeat
      Repeat
      Until Keypressed;
      Ch:=Readkey;
      Case Ch of
        #27:Luar:=True;
```

Proses penambahan brightness dilakukan dengan mengisi seluruh intensitas palette tujuan yaitu User_Pal menjadi warna putih terang. Kemudian dilakukan pemanggilan procedure METAMORPH_PALETTE untuk mengubah Active_Pal secara perlahan menjadi User_Pal. Perubahan ini tidak dilaksanakan sampai Active_Pal menjadi User_Pal. Banyaknya perubahan yang dilakukan tergantung dari jumlah dilaksanakannya proses di bawah ini (Berapa lama Anda menekan

tombol 1). Untuk itu kita tidak bisa menggunakan ACTIVE_TO_PALETTE karena procedure ini akan mengubah isi Active_Pal menjadi User_Pal sampai tuntas.

Procedure REFRESH_PALETTE berguna untuk mengaktifkan palette pada Active_Pal ke layar monitor.

```
'1':Begin
      Set_All_Palette(User_Pal,63,63,63);
      Metamorph_Palette(Active_Pal,User_Pal,1);
      Refresh_Palette;
    End;
```

Prosesnya sama dengan penambahan brightness. Seluruh intensitas User_Pal sebagai palette tujuan diisi dengan warna hitam. Kemudian dilakukan lagi pemanggilan procedure METAMORPH_PALETTE dan REFRESH_PALETTE.

```
'2':Begin
      Set_All_Palette(User_Pal,0,0,0);
      Metamorph_Palette(Active_Pal,User_Pal,1);
      Refresh_Palette;
    End;
```

Meningkatkan intensitas merah dilakukan dengan mengisi palette tujuan yaitu User_Pal dengan intensitas merah maksimum sedangkan pada intensitas biru dan hijau diberi nilai 0 sehingga kedua intensitas ini pada layar monitor nantinya akan semakin berkurang hingga mencapai 0. Demikian pula dengan proses peningkatan intensitas hijau dan biru.

```
'R','r':Begin
      Set_All_Palette(User_Pal,63,0,0);
      Metamorph_Palette(Active_Pal,User_Pal,1);
      Refresh_Palette;
    End;
'G','g':Begin
      Set_All_Palette(User_Pal,0,63,0);
      Metamorph_Palette(Active_Pal,User_Pal,1);
      Refresh_Palette;
    End;
'B','b':Begin
      Set_All_Palette(User_Pal,0,0,63);
      Metamorph_Palette(Active_Pal,User_Pal,1);
      Refresh_Palette;
    End;
```

Di sini Active_Pal diubah menjadi palette tujuan = Default_Pal. Maksudnya ialah untuk mengembalikan secara perlahan ke intensitas palette Default. Untuk itu kita tinggal melakukan pemanggilan procedure METAMORPH_PALETTE dengan mengisikan parameter Default_Pal sebagai palette tujuan.

```
        ' ':Begin
              Metamorph_Palette(Active_Pal,Default_Pal,1);
              Refresh_Palette;
            End;
      End;
    Until Luar;
```

Seperti biasa, untuk memperindah selesainya program, dilakukan peredupan gambar terlebih dahulu.

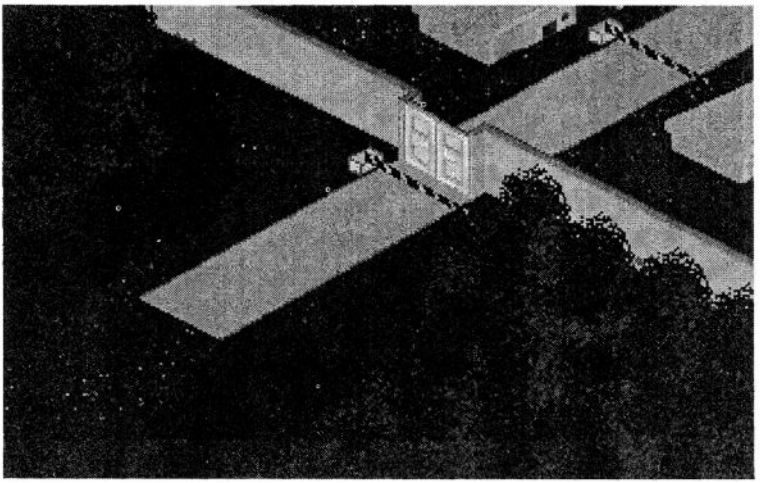
```
    Set_All_Palette(User_Pal,0,0,0);
    Active_To_Palette(User_Pal,30);
  End;
```

ANIMASI SCROLL

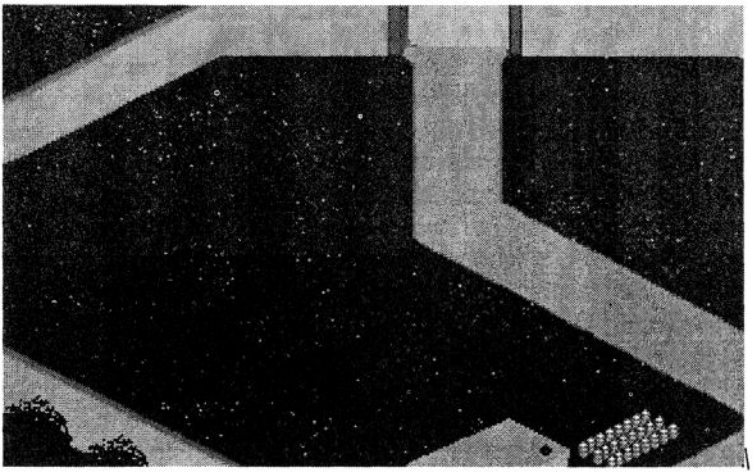
Mengingat proyek yang akan kita buat pada akhir bab ini adalah pembuatan game bertipe RAPTOR, ada teknik yang masih kurang. Pernahkan Anda memperhatikan background permainan pada waktu bermain RAPTOR? Background yang dipergunakan tidak hanya gambar diam, seperti yang program REAL1.PAS. Background yang dipergunakan adalah background yang dapat bergeser turun dari atas layar, sehingga seakan-akan kitalah yang bergerak ke arah atas layar. Hal ini mudah untuk dilakukan. VGACAD.TPU telah menyediakan fasilitas pembuatan tipe background seperti ini. Sebenarnya prinsip dari sistem ini adalah sebagai berikut. Pada memori, kita menciptakan virtual page yang besarnya melebihi ukuran page standard (lebih dari 64000 bytes). Besar memori yang harus Anda pesan adalah sejumlah 320 x Lebar gambar. Bila Anda ingin membuat gambar background yang besarnya tiga kali lipat ukuran layar normal. Anda harus memesan memori sejumlah 320 x 3 x 200 = 192000 bytes. Sebenarnya jumlah memori untuk menyimpan gambar background ini bisa dicompress dengan membuat suatu procedure untuk compress dan uncompress, tetapi dalam buku ini kita tidak akan membahasnya mengingat kita masih mempelajari dasar dari pemrograman animasi dan game profesional.

Karena besar layar yang Anda buat tiga kali lebar layar biasa, maka koordinat Y dari layar = 200 x 3 = 600. Untuk melakukan scroll dari bawah ke atas, Anda cukup melakukan counter koordinat Y dari background page yang hendak ditampilkan ke layar.

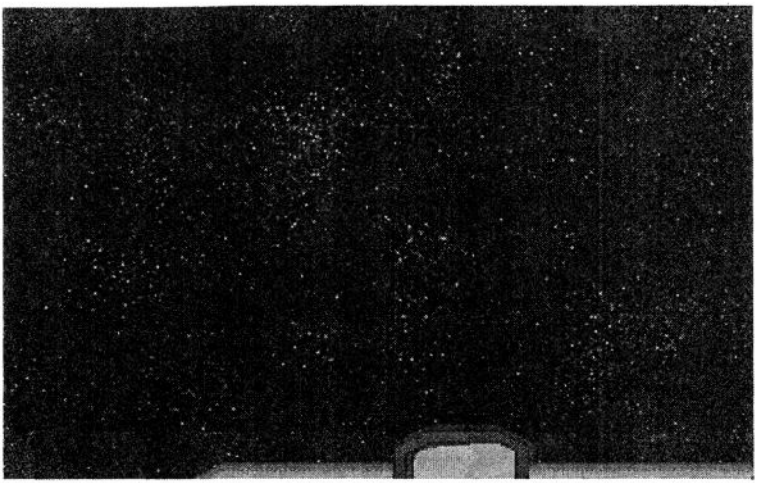
Mula-mula dilakukan duplikasi dari background page mulai dari baris 1 hingga baris 200. Empat ratus baris sisanya tidak perlu dipedulikan sehingga apa yang Anda lihat pada layar monitor adalah gambar dari background mulai baris 1 hingga 200. Langkah selanjutnya ialah melakukan duplikasi lagi dari layar background ke layar tampilan mulai baris 2 hingga baris 201 dari gambar background. Sehingga gambar yang kita lihat pada layar monitor kelihatan bergeser ke atas karena baris pertama dari layar tampilan tidak lagi menampilkan gambar background baris pertama melainkan menampilkan gambar background baris kedua. Proses ini dilakukan berulang-ulang hingga seluruh gambar background telah ditampilkan di layar.



Gambar 3.25 Layar sebelum dilakukan proses scroll



Gambar 3.26 Layar setelah dilakukan proses scroll.



Gambar 3.27 Layar setelah proses scroll lebih lanjut

Memang kelihatannya proses ini agak sulit. Apabila Anda masih bingung, coba bayangkan teman Anda berdiri di depan Anda sambil memegang bingkai lukisan. Kemudian suruhlah teman Anda yang satu lagi lagi mengambil kalender yang panjang dan melewatkan kalender tersebut di belakang bingkai mulai dari bawah ke atas hingga seluruh kalender telah melewati bingkai. Pada saat kalender bergerak amatilah gambar yang Anda lihat di dalam bingkai. Jangan mengikuti gerakkan kalender ke atas. Tetaplah memandang gambar yang ada di tengah-tengah bingkai seakan-akan Anda tidak dapat melihat gambar di luar bingkai. Seperti inilah animasi scroll bekerja. Bingkai itu mewakili layar monitor Anda sedangkan kalender yang panjang tersebut mewakili background page Anda.

Sistem scroll ini dinamakan Vertical Scroll. Sebenarnya masih ada dua macam scroll lagi yaitu Horisontal Scroll dan Parallel Scroll. Kedua Tipe scroll yang terakhir ini memerlukan teknik yang agak rumit. Pada buku ini kita tidak akan membahasnya terlebih dahulu.

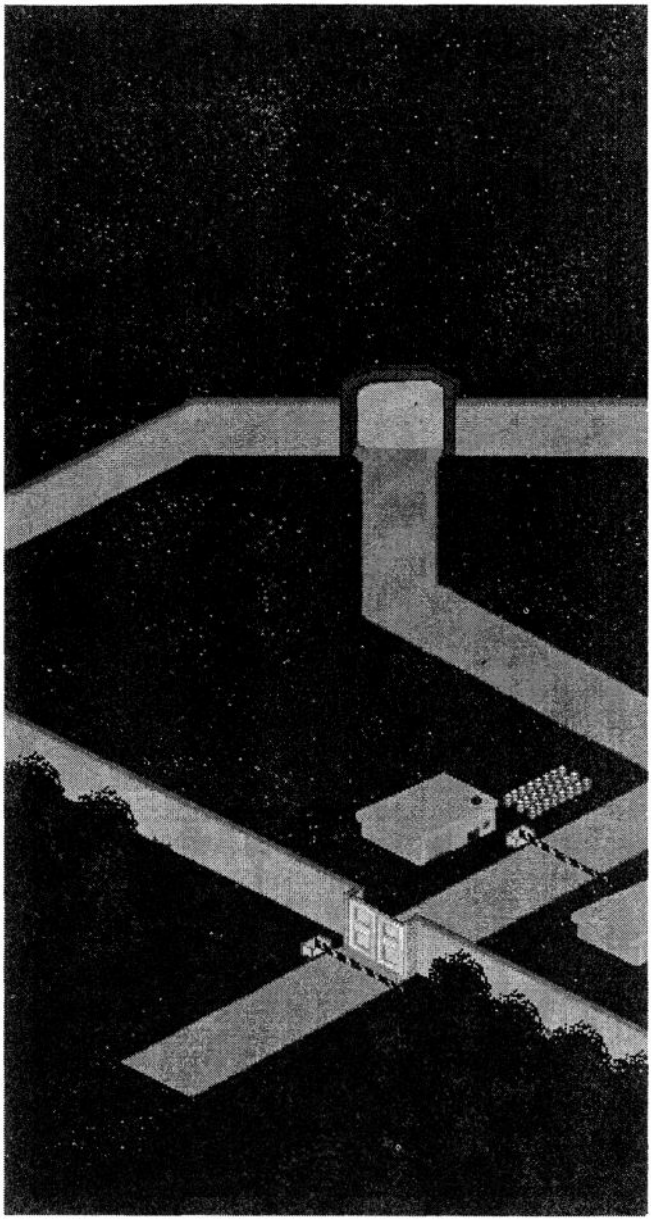
FASILITAS SCROLL PAGE PADA VGACAD.TPU

PROCEDURE LOAD SCROLL PAGE

Sintaks : Load_Scroll_Page (Nama_File);
Variabel : Nama_File = Nama File tempat background page disimpan. Besar dari file background ini maksimum tiga kali besar memori page yaitu 192000 bytes.
Fungsi : Untuk meload background page dari file ke memori.

```
Procedure Load_Scroll_Page (Name:String);
Var Size: Longint;
    Ctr: Longint;
  Begin
    Ctr:=1;
```

Pesan memori untuk setiap koordinat Y, masing-masing sebanyak 320 bytes. Totalnya ialah 192000 bytes. Dilakukan pemesanan memori untuk setiap lompatan koordinat Y dengan alasan TURBO PASCAL tidak dapat melakukan pemesanan memori lebih dari 64 kbytes pada satu variabel pointernya. Selain itu agar kita dapat melakukan proses MOVE tanpa perlu menghitung offset start dari background page yang akan dikeluarkan ke virtual page. Cukup dengan mencacah urutan array dari background page pada setiap loop. Setiap variabel counter ini bertambah, 320 titik pada background page dilompati (320 titik sama dengan satu baris). Dengan cara ini maka layar akan terlihat seperti digulung. Pada memori komputer akan dibentuk background page seperti berikut.



```
For A:=1 to 600 do
      GetMem(Page_Scroll[A],320);
```

Buka file background.

```
    Assign(File_P,Name);
    Reset(File_P,1);
    Size:=FileSize(File_P);
```

Hitung lebar dari gambar background (lebar gambar = jumlah koordinat Y). Rumusnya ialah besar dari file dibagi dengan 320. Karena panjang gambar adalah 320 titik setiap 1 koordinat Y.

```
    Page_Width:=Size div 320;
```

Baca data gambar satu per satu.

```
    While Size>64000 do
      Begin
        BlockRead(File_P,Page_Scroll[Ctr]^,64000);
        Size:=Size-64000;
        Inc(Ctr,200);
      End;
    BlockRead(File_P,Page_Scroll[Ctr]^,Size);
```

Tutup file gambar.

```
    Close(File_P);
  End;
```

PROCEDURE FREE SCROLL PAGE

Sintaks : Free_Scroll_Page;
Fungsi : Untuk menmbersihkan kembali memori yang dipergunakan untuk background page.

```
Procedure Free_Scroll_Page;
Var Size: Word;
  Begin
    For A:=1 to 600 do
      FreeMem(Page_Scroll[A],320);
  End;
```

PROCEDURE PUT SCROLL PAGE

Sintaks : Put_Scroll_Page (PicBuff,Y);

Variabel : PicBuff = Virtual memori tempat background page akan ditampilkan.

Y = Start pembacaan gambar di background page mulai baris ke Y.

Fungsi : Untuk meletakkan background page pada virtual memori. Di sini Anda dapat memasukkan variabel Y sebagai parameter yang menunjukkan baris awal dari background page yang akan ditampilkan ke layar. Bila Anda isikan 6 maka gambar pada background page akan dilompati 5 baris = 5 x 320 = 1600 bytes. Gambar yang akan ditampilkan adalah mulai baris ke 6 sampai baris ke 205.(Lebar gambar yang ditampilkan selalu 200 baris).

```
Procedure Put_Scroll_Page (Var PicBuff; Y:Integer);
  Begin
```

Koordinat Y harus lebih besar dari 1 dan juga tidak boleh melebihi lebar gambar (jumlah koordinat Y).

```
    If (Y<=Page_Width) and (Y>=1) then
      Begin
```

Jika sisa lebar gambar masih lebih besar dari 200, langsung dilakukan pemindahan data gambar dari background page baris ke Y sejumlah 32000 words = 64000 bytes = 200 baris.

```
        If Page_Width-Y>=200 then
          MoveW(Page_Scroll[Y]^,PicBuff,32000)
        Else Begin
```

Jika ternyata dengan start awal = Y, lebar layar yang dihasilkan tidak dapat berjumlah 200 baris maka dilakukan proses peletakan sebagian. Situasi ini bisa terjadi, misalnya background page Anda sebesar 600 baris. Anda memasukkan parameter Y = 500. Ini berarti Anda hendak meletakkan gambar mulai baris ke 500 hingga 700. Hal ini tentu saja tidak mungkin mengingat lebar gambar cuma 600 baris. Untuk itu dilakukan peletakan sebagian. Yaitu baris ke 500 hingga 600 saja. Sisa layar yang separuhnya diisi dengan background page mulai baris awal lagi.

Teknik ini sering digunakan pada game-game bertipe RAPTOR. Apabila Anda bermain RAPTOR, coba perhatikan background yang bergeser, setelah beberapa lama, background akan diulang lagi. Tetapi karena background tersebut ditumpuk oleh object darat (bangunan-bangunan, tank dan lain-lain), teknik ini tidak tampak. Yang perlu Anda ingat ialah, object-object darat tersebut bukanlah bagian dari background. tetapi sprite yang pada process page diletakkan di atas gambar background. Lebar bagian gambar yang ditaruh pada virtual page dapat dicari dengan rumus Lebar_gambar-Y +1 (Pada situasi di atas = 600-500+1=101). Sehingga jumlah memori yang diperlukan adalah (Lebar_Gambar-Y + 1)*160 words = (Lebar_Gambar-Y + 1)*320 bytes.

```
MoveW(Page_Scroll[Y]^,PicBuff,(Page_Width-Y+1)*160);
```

Jika masih ada sisa virtual page yang belum terisi maka kerjakan peletakan bagian gambar mulai dari baris awal untuk mengisi sisa virtual page. Untuk itu dipanggil procedure PUT_PART.

```
        If Page_Width-Y+1<>200 then
       Put_Part(Page_Scroll[1]^,PicBuff,Page_Width-Y+1,200-(Page_Width-Y+1));
      End;
    End;
End;

Procedure Put_Part (Var Page_Scroll; Var PicBuff; Y,Width: Integer); ASSEMBLER;
  Asm
    Push ds                 { Push Stack                          }
```

Arahkan ES:DI ke Virtual page.

```
    Les di,PicBuff          { Arahkan ES:DI ke PICBUFF            }
```

Arahkan DS:SI ke background page.

```
    Lds si,Page_Scroll      { Arahkan DS:SI ke Page_Scroll        }
```

Hitung offset awal peletakan gambar pada virtual page.

```
    Mov dx,Y                { Dx = Y awal                         }
    Mov ax,dx               { Ax = Dx                             }
    Shl dx,1
    Shl dx,1                { Dx = Dx * 4                         }
```

```
    Add dx,ax               { Dx = Dx + Ax                  }
    Shl dx,1
    Shl dx,1
    Shl dx,1
    Shl dx,1
    Shl dx,1
    Shl dx,1                { Dx = Dx * 32                  }
    Add di,dx               { Offset = Offset + Dx          }
```

Hitung besar data yang hendak diduplikasi ke virtual page. Di sini cukup dilakukan perkalian hingga sampai x 16 karena akan dilakukan pemindahan data secara WORD.

```
    Mov dx,Width            { Dx = Lebar sisa layar         }
    Mov ax,dx               { Ax = Dx                       }
    Shl dx,1
    Shl dx,1                { Dx = Dx * 4                   }
    Add dx,ax               { Dx = Dx + Ax                  }
    Shl dx,1
    Shl dx,1
    Shl dx,1
    Shl dx,1
    Shl dx,1                { Dx = Dx * 16                  }
```

Isikan CX sebagai counter sejumlah DX.

```
    Mov Cx,Dx               { Counter Cx sejumlah Dx        }
```

Pindahkan gambar dari DS:SI ke ES:DI.

```
    Rep MovsW               { Pindahkan DS:SI ke ES:DI      }
    Pop ds                  { Pop Stack                     }
  End;
```

PEMANFAATAN FASILITAS SCROLL PAGE PADA VGACAD.TPU

Bagaimana cara memakainya? Tentu saja sebelum kita melakukan loading gambar background page, kita harus menciptakan gambarnya terlebih dahulu. Untuk itu kita dapat mempergunakan software lain. Disarankan menggunakan DELUXE PAINT. Di sini Anda dapat memiliki

area untuk menggambar jauh lebih besar dari layar monitor. Bila Anda memiliki software ini atau software lain yang dapat memperluas area gambar, coba untuk mengubah setting layar menjadi 320 x 600. Kemudian buatlah gambar background yang Anda sukai. Sebelum Anda masuk ke software graphic Anda, aktifkan terlebih dahulu program CAP2.EXE kita.

Setelah gambar selesai Anda rancang, kini tinggal melakukan proses capture. Pindahkan posisi layar ke tempat yang paling atas kemudian lakukan capture. Lalu geserlah gambar sebanyak 200 pixel ke bawah dan lakukan capture lagi. Demikian seterusnya sehingga seluruh gambar telah diambil. Keluarlah dari program. Bila proses yang Anda lakukan benar, Anda akan memiliki beberapa file gambar dan beberapa file palette yang merupakan hasil dari proses capture yang telah Anda lakukan. Procedure LOAD_SCROLL_PAGE tidak dapat melakukan pembacaan gambar background yang terpisah pada beberapa file. Untuk itu beberapa file ini perlu Anda gabung menjadi satu file. Untuk menggabungkannya Anda dapat menggunakan fungsi COPY dari DOS dengan parameter penambahan. Juga bisa dengan membuat program khusus untuk melakukan penggabungan. Misalnya sebagai berikut.

Listing Program

```
{-------------------------------------------------------------}
{ File      :  Join.Pas                                       }
{ Compiler  :  Turbo Pascal 7.00                              }
{ Fungsi    :  Untuk menciptakan background scroll dari beberapa gambar.  }
{-------------------------------------------------------------}
{ Program JOIN.PAS }
Uses Crt,Dos,Vgacad;
Var A,B:Integer;
    Fil:File;

Begin
  initVga256(dapalette);
  Assign(Fil,'Scroll.Pic');
  Rewrite(Fil,1);
  Load_Picture('1.Pic',Screen^);
  BlockWrite(Fil,Screen^,64000);
  Load_Picture('2.Pic',Screen^);
  BlockWrite(Fil,Screen^,64000);
  Load_Picture('3.Pic',Screen^);
  BlockWrite(Fil,Screen^,64000);
  Close(Fil);
End.
```

Penjelasan Listing Program

Program ini akan menggabungkan tiga file gambar hasil capture menjadi satu file dengan nama SCROLL.PIC (Ingat, jumlah file gambar yang ditambahkan tidak perlu selalu tiga buah, bila background scroll Anda hanya sebesar dua file gambar, Anda tidak perlu melakukan proses LOAD_PICTURE dan BLOCKWRITE untuk file gambar nomor 3).

Setelah Anda menjalankan program ini, seharusnya file SCROLL.PIC telah terbuat. Sedangkan untuk palette, Anda tinggal memilih satu diantara empat file palette yang dibentuk oleh program CAP2.EXE. Jelas keempat file tersebut identik karena Anda melakukan proses capture pada satu gambar yang sama. File SROLL.PIC dan SCROLL.PAL ini sudah ada pada direktori Anda pada saat Anda menginstall diskette yang disertakan pada buku ini. Dengan cara di ataslah file ini saya buat. Sekarang cobalah program di bawah ini.

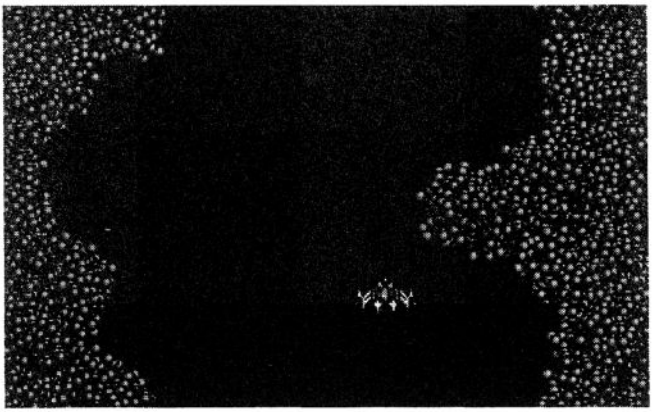
Listing Program

```
{-------------------------------------------------------------}
{ File     :  Scroll1.Pas                                     }
{ Compiler :  Turbo Pascal 7.00                               }
{ Fungsi   :  Untuk melakukan proses scroll dengan background ruang angkasa.}
{-------------------------------------------------------------}

Uses Crt,Dos,VgaCad;

Var A,B:Integer;
    Fil:File;

Begin
  OpenVga256;
  Install_Palette('Scroll.Pal');
  Load_Scroll_Page ('Scroll.Pic');
  Repeat
    For A:=600 downto 1 do
      Begin
        Put_Scroll_Page  (Screen^,A);
      End;
  Until Keypressed;
  CloseVga256;
End.
```

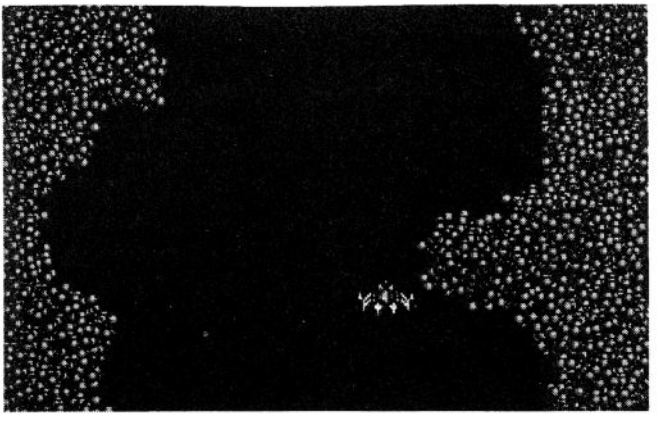
Penjelasan Listing Program

Pada program di atas dilakukan pemanggilan PUT_SCROLL_PAGE dengan parameter variabel A yang diloop dari bawah ke atas (dari koordinat baris 600 hingga baris 1).

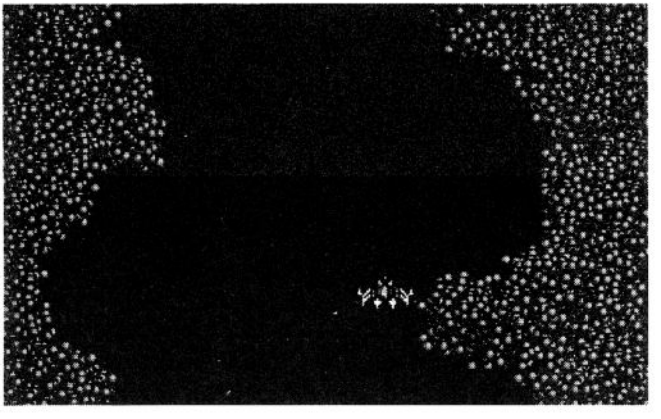
Sebelumnya sudah kita bahas bahwa variabel baris (pada situasi ini ialah variabel A) akan kita inputkan pada parameter menginstruksikan bahwa letakkan gambar mulai baris ke A dari background page ke virtual memori. Sehingga bila variabel A ini kita cacah maju (dari 1 ke 600) maka start baris yang akan ditampilkan akan bergeser ke bawah sehingga seakan-akan layar digeser ke atas. Apa yang terjadi kalau Anda meletakkan gambar pesawat yang tidak bergerak pada tengah-tengah layar? Pesawat akan terlihat bergerak ke bawah, karena background bergeser ke arah atas.



Gambar 3.28 Pesawat diletakkan pada background yang discroll. Perhatikan posisi pesawat pada saat ini. Pesawat masih belum melewati tonjolan dinding di sebelah kanan.



Gambar 3.29 Pada saat ini background telah digulung lebih lanjut. Walaupun posisi pesawat pada layar monitor tidak berpindah, pesawat terlihat bergerak mendekati tonjolan di sebelah kanan dinding.



Gambar 3.30 Pesawat telah hampir melewati tonjolan dinding karena dilakukan proses scroll lebih lanjut.

Untuk memeriksanya, gantilah looping di atas menjadi FOR A:=1 to 600 do dan lihat apa yang terjadi bila Anda membayangkan ada sebuah pesawat pada tengah-tengah layar. Pesawat itu akan terlihat bergerak

mundur (ke bawah layar). Untuk mengatasi masalah ini, kita tidak perlu mengganti proses pada procedure PUT_SCROLL_PAGE kita. Cukup parameter yang akan kita masukkan sebagai start baris saja yang kita balik. Untuk itulah digunakan loop dengan arah yang terbalik.

Pemakaian animasi scroll ini dapat Anda variasi untuk berbagai macam program animasi. Diantaranya ialah membuat gambar besar yang dapat Anda geser dari atas ke bawah ataupun sebaliknya dengan menggunakan tombol kursor. Hal ini tidaklah sulit untuk dilakukan. Cobalah program di bawah ini.

Listing Program

```
{-------------------------------------------------------------------}
{ File       :  Scroll2Pas                                          }
{ Compiler   :  Turbo Pascal 7.00                                   }
{ Fungsi     :  Untuk melakukan proses scroll yang dapat Anda kendalikan }
{-------------------------------------------------------------------}


Uses Crt,Dos,VgaCad;

Const Up   = 1;
      Down = 2;

Var A,B:Integer;
    Fil:File;
    Dir:Byte;
    Ch:Char;
    Luar:Boolean;

Procedure Initialize;
  Begin
    Luar:=False;
    A:=400;
    Dir:=0;
    Install_Palette('Scroll2.Pal');
    Load_Scroll_Page ('Scroll2.Pic');
  End;

Procedure Main;
  Begin
    Repeat
      Repeat
        Put_Scroll_Page(Screen^,A);
        If Dir=Up then Dec(A);
```

```
        If Dir=Down then Inc(A);
        If A<1 then A:=599+A;
        If A>600 then A:=A-600;
      Until Keypressed;
      Ch:=Readkey;
      Dir:=0;
      Case Ch of
        #72:Dir:=Down;
        #80:Dir:=Up;
        '2':Dec(A,15);
        '8':Inc(A,15);
        #27:Luar:=True;
      End;
    Until Luar;
  End;

Begin
  OpenVga256;
  Initialize;
  Main;
  CloseVga256;
End.
```

Penjelasan Listing Program

Program di atas akan menampilkan gambar Wu Kong. Gambar tersebut tidak dapat ditangkap seluruhnya oleh layar monitor. Untuk itu Anda dapat menggeser gambar dengan menggunakan kursor atas dan bawah. Tombol '8' dan '2' dapat Anda gunakan untuk menggeser gambar secara cepat. Sedangkan untuk keluar dari program gunakan tombol ESC.

Sebenarnya sistem pemrogramannya sama saja dengan program SCROLL1.PAS. Di sini kita menambahkan penekanan tombol dan dengan instruksi lewat tombol tersebut selanjutnya dilakukan perubahan start baris gambar background yang hendak ditampilkan di layar monitor. Variabel yang menunjukkkan start baris adalah variabel A.

Seperti biasa pada awal program dipanggil procedure INITIALIZE. Di sini dilakukan pembacaan file palette sebagai palette default dan juga file background page. Variabel A yang menunjukkan start baris awal yang hendak ditampilkan diisi dengan 400 (Berarti gambar yang akan ditampilkan mulai baris 400 hingga 599). Perlu diketahui bahwa file Scroll2.Pic ini besarnya adalah 3 x memori page yaitu = 64000 x 3 bytes. Sehingga lebar background page adalah 600.

```
Procedure Initialize;
  Begin
    Luar:=False;
    A:=400;
    Dir:=0;
    Install_Palette('Scroll2.Pal');
    Load_Scroll_Page ('Scroll2.Pic');
  End;
```

Langkah berikutnya ialah memanggil procedure MAIN.

```
Procedure Main;
  Begin
    Repeat
      Repeat
```

Letakkan background page mulai baris ke A hingga A+199 ke visual page.

```
        Put_Scroll_Page(Screen^,A);
```

Jika instruksi adalah UP maka kurangi start baris.

```
        If Dir=Up then Dec(A);
```

Jika instruksi adalah DOWN maka tambah start baris.

```
        If Dir=Down then Inc(A);
```

Deteksi apakah variabel A telah berada di luar jangkauan batas layar (Jangkauan batas layar = 1 hingga 600). Jika Ya, bila variabel A < 0 maka ulang lagi mulai 600. Jika variabel A > 600 ulang variabel A mulai dari 1. Dengan sistem ini maka apabila background page telah melebihi jangkauan layar, gambar pada baris awal akan diulang lagi.

```
        If A<1 then A:=599+A;
        If A>600 then A:=A-600;
      Until Keypressed;
      Ch:=Readkey;
      Dir:=0;
```

Deteksi tombol keyboard.

```
    Case Ch of
      #72:Dir:=Down;
      #80:Dir:=Up;
      '2':Dec(A,15);
      '8':Inc(A,15);
      #27:Luar:=True;
    End;
  Until Luar;
End;
```

SELAMAT ! Dengan ini Anda telah menguasai dasar-dasar pemrograman animasi dan game profesional ! Pada bab-bab selanjutnya kita akan mulai mempelajari bagaimana merakit sebuah game dari nol hingga menjadi benar-benar sempurna.

BAB 4

PEMBUATAN GAME PROFESIONAL

PENDAHULUAN

Pada bab ini diharapkan Anda telah benar-benar menguasai dasar-dasar pemrograman game yang telah dibahas pada bab-bab sebelumnya.

Untuk membuat sebuah game, kita tidak hanya dituntut menguasai teknik animasi. Kita harus dapat menerapkan ilmu-ilmu matematika bahkan fisika yang telah kita pelajari. Memang kebanyakan ilmu-ilmu tersebut membosankan karena hanya teori melulu yang diajarkan. Setelah Anda menerapkannya pada pembuatan game, ilmu yang biasanya membosankan tersebut berubah menjadi sangat mengasyikkan.

Anda semua tentu tahu tentang matriks. Bila mengingat matriks, kita pasti teringat akan bilangan-bilangan yang disusun dalam kotak yang dikalikan, dicari inversnya, dicari determinannya dan lain-lain. Ilmu matriks ini sangat berguna pada pembuatan game terutama pada pembuatan game bertipe simulator 3D seperti F19, F117A, Flight Simulator. Di sini matriks digunakan untuk melakukan proses rotasi, pembesaran object maupun pemindahan posisi object. Bahkan game WING COMMANDER yang sangat tersohor itu juga memakai perhitungan matriks. Kita akan dapat membuat game setaraf dengan WING COMMANDER tapi tidak pada edisi buku ini (mungkin pada edisi berikutnya) karena teknik pemrogramannya sangat sulit.

Contoh lainnya yaitu DOOM (Anda pasti sudah familiar dengan game canggih yang satu ini). DOOM memiliki tingkat pemrograman yang tersulit di antara game-game yang telah saya pelajari. Di sini dipergunakan rumus-rumus trigonometri dan juga perhitungan fisika untuk kalkulasi RAY TRACING (penelusuran sinar) serta perhitungan focus mata. Dengan gabungan antara teori FISIKA dan MATEMATIKA itu dihasilkan game yang bersudut pandang orang pertama.

PEMBUATAN PROYEK GAME PROFESIONAL

WELCOME TO THE CLUB! Kita akan membuat game profesional. Game bertipe RAPTOR adalah game yang memiliki teknik yang termudah di antara game-game lainnya seperti game 3D dan ADVENTURE. Selain itu tidak diperlukan gambar yang terlalu banyak mengingat kita baru mempelajari dasar-dasar teknik pemrograman animasi dan game.

Untuk membuat sebuah game bertipe RAPTOR ini, ada beberapa tahap yang harus kita lakukan.

- Perencanaan (Design).
- Pembuatan Gambar.
- Pembuatan Editor (Tidak selalu diperlukan).
- Pembuatan Engine (Program).
- Pembuatan Data dan Level.
- Pembenahan Akhir.

PERENCANAAN

Pada saat ini kita harus merencanakan game seperti apa yang hendak kita buat. Misalnya saja sekarang, kita hendak membuat game pesawat. Situasinya ialah kita akan menerbangkan pesawat, dimana kita diharuskan menghancurkan musuh-musuh yang keluar dari berbagai sisi layar. Pesawat kita dapat bergeser horizontal maupun vertical selama masih dalam batas layar. Pada saat yang sama background harus digerakkan dari atas ke bawah sehingga menimbulkan efek berjalan.

Kita juga harus merencanakan gambar apa saja yang kita perlukan dalam membuat game ini. Pertama tentu saja gambar pesawat, peluru,

ledakan, background dan sebagainya. Dan tentu saja sebuah editor program untuk merakit level.

PEMBUATAN GAMBAR

Pada umumnya, di sinilah letak masalah dari para programmer. Karena belum tentu orang yang ahli memprogram, ahli juga dalam menggambar. Biasanya team-team pembuat game seperti SIERRA, LUCAS ART, ORIGIN mempekerjakan orang-orang yang memiliki keahlian yang berbeda-beda. Sebagian bertugas untuk menyusun scenario game, sebagian bertugas membuat program dan sisanya ditugaskan untuk menggambar (sebagai graphic artist).

Untuk menghasilkan gambar, kita dapat mempergunakan bantuan dari software graphic. Adapun software graphic yang saya pakai adalah Deluxe Paint II buatan Electronic Arts. Software ini dirancang khusus untuk pembuatan game. Di sini Anda dapat menggabung-gabungkan gambar Anda menjadi serangkaian animasi sehingga Anda dapat mencobanya terlebih dahulu sebelum Anda capture untuk dipergunakan dalam program animasi Anda.

PEMBUATAN EDITOR

Editor merupakan program yang dapat Anda pergunakan untuk mengisikan data yang diperlukan. Contohnya data level yang meliputi kapan sebuah pesawat harus dikeluarkan, apa senjata yang dimiliki pesawat tersebut, kapan senjata itu ditembakkan, berapa power yang dimiliki pesawat tersebut dan lain sebagainya. Sebenarnya data ini dapat Anda masukkan melalui program. Tapi hal ini tidak efisien. Selain program Anda menjadi sangat besar, untuk penambahan data baru diperlukan perubahan pada main program. Bayangkan saja kalau jumlah pesawat musuh ada seratus jenis. Anda harus memasukkan data power, senjata dan aktifitas dari tiap pesawat, program Anda akan bertambah ratusan baris.

Sekarang pertanyaannya, siapa yang harus membuat editor ini? Tentu saja Anda sendiri, karena Anda yang hendak membuat game.

Editor ini harus Anda rancang sedemikian rupa sehingga mudah untuk digunakan dan juga dilengkapi dengan display sehingga untuk merancang sebuah level tidaklah terlalu sulit.

Keuntungan lain yang Anda dapat dari pembuatan editor ini adalah, Anda dapat menyuruh orang lain yang belum tentu bisa memprogram untuk merancang sebuah level. Jadi tugas Anda hanya membuat hal-hal yang berkenaan dengan program.

PEMBUATAN ENGINE

ENGINE sebenarnya adalah program dari game Anda sendiri. Tentu saja sebelum membuat program yang sesungguhnya Anda harus membuat rencananya terlebih dahulu. Teknik apa saja yang hendak Anda pergunakan. Satu hal yang perlu Anda ingat, buatlah UNIT apabila memungkinkan dan usahakan agar UNIT tersebut dapat Anda pergunakan untuk membuat game-game yang lain (Contohnya seperti VGACAD.TPU). Sehingga bila Anda hendak membuat game lagi, Anda tidak perlu mulai dari dasar.

PEMBUATAN DATA DAN LEVEL

Pembuatan data dan level ini dapat Anda lakukan seiring dengan pembuatan engine. Karena dalam memprogram game engine, Anda tentu harus mencobanya untuk mengetahui apakah game engine itu berjalan dengan benar atau tidak. Tentu saja paling tidak Anda telah membuat data satu level terlebih dahulu agar game engine Anda bisa jalan.

PEMBENAHAN AKHIR

Apabila semuanya telah beres, program Anda telah jalan dengan sempurna, semua level telah Anda buat, bagian terkhir tinggal pembenahan. Di sini Anda dapat menambahkan suara, musik (pakai sound blaster ataupun pakai real sound), Anda juga dapat merancang intro, menu dan akhir dari game yang Anda buat.

BAB 5

PEMBUATAN GAME FIRE FIGHTER

PERENCANAAN

Sekarang kita menerapkan tahap-tahap teknik pembuatan game di atas pada pembuatan game FIRE FIGHTER ini. Tahap ini benar-benar saya lakukan pada saat membuat game ini.

Game yang akan kita buat ini adalah game *arcade* dengan menggunakan vertical scrolling. Anda akan mengendalikan pesawat yang divisualisasikan dari atas (top view) sebagai musuh akan ditampilkan berbagai macam pesawat dari berbagai posisi layar yang akan menyerang Anda. Selama permainan, layar akan digeser secara vertikal.

Dan yang paling penting adalah memberi judul, game ini kita beri nama FIRE FIGHTER.

- Pesawat Anda.
  - Dapat digerakkan secara vertikal maupun horisontal selama masih dalam layar permainan.
  - Dapat melakukan penembakan. Jenis tembakan ditentukan dengan jumlah weapon power yang Anda ambil.
  - Dapat melakukan penembakan bomb. Pada saat bomb dijatuhkan, semua musuh yang ada di layar akan terkena tembakan sehingga powernya akan terus berkurang. Selain itu, semua tembakan musuh harus dilenyapkan. Jumlah bomb maksimum adalah 5.
  - Memiliki power dan akan meledak apabila power habis. Jumlah nyawa cadangan adalah tiga buah.
  - Gambar dari pesawat Anda akan terdiri dari beberapa sprite yang terus diulang pada saat permainan sehingga menimbulkan effek hidup (Pada beberapa sprite ini dilakukan perubahan gambar after burner/api dari pesawat sehingga terlihat hidup). Animasi yang dihasilkan akan lebih bagus, coba bandingkan dengan program kita yang terdahulu di mana kita dapat menggerakkan pesawat secara vertikal maupun horisontal (REAL1.PAS). Pada program REAL1.PAS itu sprite yang ditampilkan cuma satu sehingga animasi terlihat kaku.

- Pesawat musuh.
  - Setiap pesawat musuh memiliki route tertentu, di mana dia akan muncul di layar monitor, bergerak beberapa saat, mungkin melakukan serangan kemudian menghilang dari layar.
  - Posisi start dari route sebuah pesawat musuh dapat berada di mana saja, dapat ditentukan pada saat merancang level dengan editor.
  - Posisi selanjutnya juga bebas, dapat ditentukan pada saat merancang level dengan editor.
  - Musuh dapat melakukan tembakan kapan saja selama dia masih hidup tapi dibatasi satu tembakan setiap frame-nya (Maksudnya hanya bisa menembak sekali setiap satu framenya, nggak bisa sekaligus dua tembakan).
  - Tiap musuh dapat memiliki senjata yang berbeda-beda baik gambarnya, daya rusaknya dan juga tipe senjatanya (Senjata yang bisa mengejar atau tidak).
  - Tiap musuh memiliki power yang berbeda-beda dan akan hancur bila power ini habis.
  - Gambar dari pesawat musuh ini memiliki teknik yang sama dengan gambar dari pesawat Anda yaitu menggunakan beberapa sprite yang diulang.

- Senjata musuh.
  - Tipe senjata ada beberapa macam diantaranya ialah :
    - Senjata yang punya arah yang tetap (Tidak dapat berubah arah).
    - Senjata yang dapat mengejar secara horizontal (Cuma bisa berubah arah secara horizontal ke arah pesawat Anda sedangkan arah vertikal sudah pasti ke bawah).
    - Senjata yang pada saat ditembakkan, mendeteksi letak pesawat Anda sehingga senjata tersebut mengarah ke pesawat Anda (Tetapi setelah ditembakkan tidak dapat berubah arah).
  - Tiap senjata memiliki daya rusak yang berbeda-beda.
  - Gambarnya juga terdiri dari beberapa sprite yang diulang.

- Status Display.
    - Pada bawah layar akan ditampilkan bar power (untuk menunjukkan jumlah power Anda), jumlah bomb yang masih Anda miliki dan juga jumlah nyawa cadangan.

- Animasi Object.
    - Animasi ledakan akan ditampilkan apabila sebuah pesawat meledak, juga pada saat bomb Anda jatuhkan.
    - Animasi pecahan pesawat dan percikan api akan ditampilkan apabila sebuah pesawat terkena tembakan.
    - Gambarnya terdiri dari beberapa sprite yang tidak diulang karena ledakan, percikan api dan pecahan pesawat cuma terjadi sekali saja setelah itu lenyap, lain dengan pesawat musuh yang harus menyelesaikan routenya kemudian lenyap dari layar.

- Bonus.
    - Object bonus ditampilkan pada saat-saat tertentu. Adapun jenis-jenis bonus tersebut adalah sebagai berikut
        - Bonus Nyawa yang akan memberi tambahan nyawa cadangan.
        - Bonus Power Up yang akan memperbarui senjata Anda.
        - Bonus Bomb yang akan akan menambah jumlah bomb Anda.
    - Gambarnya terdiri dari beberapa sprite yang diulang.

Dengan rencana ini paling tidak kita telah memiliki gambaran game seperti apa yang harus kita buat.

PEMBUATAN GAMBAR

Dari rencana di atas, tentunya kita dapat mengetahui gambar apa saja yang perlu kita buat. Pertama tentu saja gambar background yang besarnya akan kita buat tiga layar = 3 x 64000 = 192000 bytes. Selanjutnya ialah gambar-gambar pesawat musuh, pesawat kita, senjata, ledakan, percikan api, pecahan pesawat dan gambar bonus.

Setelah itu lakukan proses capture menggunakan program CAP2.PAS yang telah kita pelajari pada bab-bab terdahulu.

Tentu saja hasil dari program capture ini adalah data gambar sebesar satu layar penuh (tidak bisa lebih dari satu layar). Maka dari itu untuk layar background kita lakukan capture secara terpisah (lakukan capture pada 200 baris pertama, kemudian 200 baris berikutnya seperti yang telah dijelaskan terdahulu, demikian seterusnya) karena besarnya tiga kali layar biasa. Deluxe Paint II menyediakan fasilitas penyuntingan gambar yang lebih besar dari ukuran resolusi monitor 320 x 200 sehingga untuk pembuatan background tidak ada masalah, kita dapat menggeser layar tersebut ke atas dan melakukan penggambaran untuk 200 baris berikutnya.

Apabila software yang Anda gunakan tidak mendukung penyuntingan gambar yang melebihi resolusi layar, Anda dapat melakukannya dengan menggambar layar background tersebut secara terpisah. Setelah itu lakukan capture pada masing-masing gambar secara berurutan.

FIRE FIGHTER MISSION EDITOR

PENDAHULUAN

Sekarang tahap pemrograman dimulai. Sebenarnya membuat editor, sama saja dengan membuat kerangka dari game engine itu sendiri. Karena struktur variabel yang hendak Anda pergunakan pada game engine sebagian besar sama dengan struktur variabel pada editor. Apalagi kalau editor itu Anda beri fasilitas PREVIEW (untuk melakukan simulasi dari level yang telah Anda sunting). Di sini animasi akan dijalankan sebagaimana halnya permainan pada game engine, bedanya Anda tidak dapat menggerakkan pesawat Anda. Cobalah jalankan FIRE FIGHTER MISSION EDITOR pada disket yang dilampirkan pada buku ini. Editor tersebut telah memiliki fasilitas PREVIEW. Dengan fasilitas ini, kita dapat mencoba level yang sedang kita sunting tanpa perlu melalui game enginenya secara langsung.

Sebelum kita membahas pembuatan editor dari FIRE FIGHTER ini, ada beberapa hal penting yang perlu kita perbuat. Pertama, mengenai gambar background yang telah Anda capture. Apabila proses yang Anda lakukan benar, Anda akan memiliki tiga gambar berukuran 64000 bytes. Masing-masing berisikan bagian dari background. Anda tentu masih ingat pada bab yang lalu, kita telah membahas mengenai scroll layar. Karena UNIT yang hendak kita pakai untuk membuat game ini nantinya ialah VGACAD.TPU, maka mau tidak mau untuk meload gambar background digunakan procedure LOAD_SCROLL_PAGE. Sedangkan procedure ini hanya dapat meload gambar yang terdapat pada satu file bukannya tiga file terpisah. Untuk itu kita perlu membuat sebuah program sederhana untuk menggabungkan keempat file gambar hasil capture menjadi satu file berukuran 192000 bytes.

Kedua, mengenai gambar pesawat, senjata dan lain-lainnya. Semua itu berserakan pada satu layar, sehingga pada awal program untuk menciptakan sprite kita terpaksa harus melakukan proses GETIMAGE satu-persatu dengan menginputkan koordinat dari gambar-gambar tersebut. FIRE FIGHTER ini memiliki sekitar 130 sprite. Kalau

kita masih ingin melakukannya dengan GETIMAGE, kita harus melakukan 130 kali GETIMAGE. Pemecahannya adalah dengan membuat sebuah program CAPTURE baru yang dapat melakukan capture hanya pada bagian yang ingin kita capture. Apabila program ini diaktifkan, yaitu dengan menekan tombol Shift + PrintScreen, akan terlihat persegi panjang pada layar monitor yang dapat Anda gerakkan maupun dapat Anda ubah besarnya. Untuk mengubah posisi titik kiri atas dari persegi panjang tekan tombol Spacebar setelah itu gunakan kursor. Sedangkan untuk mengubah titik kanan bawah, tekan tombol Spacebar lagi kemudian gunakan kursor. Anda dapat menggantikan kursor dengan tombol 2,8,6,4 untuk menghasilkan lompatan yang lebih besar. Daerah di dalam persegi panjang itulah yang nantinya akan Anda disimpan ke dalam file apabila Anda menekan ENTER. Prosesnya ialah dengan mengadakan GETIMAGE pada daerah tersebut sehingga terbentuk sebuah sprite kemudian sprite tersebut disimpan ke dalam file. Tentu saja struktur data yang dihasilkan sama dengan program 3SCR2.PAS kita yang terdahulu di mana kita menciptakan file PANAH.SPR.

Dengan program capture baru ini Anda dapat melakukan capture berkali-kali dan menyimpan hasil capture dalam sebuah file besar. Jadi bukannya setiap sprite (setiap Anda menekan Enter) dibuatkan satu file tersendiri, melainkan seluruh sprite hasil capture akan disimpan pada sebuah file. Anda dapat keluar dari program ini dengan menekan tombol ESC. File yang dihasilkan tetap sama yaitu 1.PIC dan 1.SZS.. Apabila Anda ingin melakukan proses capture lagi Anda dapat menekan tombol Shift + PrintScreen lagi dan mulai melakukan capture dan akhiri dengan tombol ESC. File yang dihasilkan adalah 2.PIC dan 2.SZS. Demikian seterusnya.

File yang berekstensi PIC berisikan sprite-sprite tersebut sedangkan file yang berekstensi SZS berisikan informasi mengenai panjang, lebar dan ukuran dari masing-masing sprite yang disimpan pada file PIC.

Muncul masalah baru, bagaimana kalau kita hendak menggabungkan semua sprite pada satu file padahal sprite-sprite tersebut tidak cukup bila ditampung pada satu layar? Anda dapat melakukan capture secara terpisah sehingga sprite-sprite tersebut akan ditampung pada beberapa file. Setelah itu Anda harus membuat program untuk menggabungkan file-file tersebut, baik yang berekstensi PIC maupun SZS.

Nama program capture yang baru itu adalah GCAP10.EXE (ada pada disket yang menyertai buku ini). Cara pemanggilan program resident ini sama dengan program CAP2.EXE yang disertai dengan parameter directory dimana file PIC dan SZS akan dibuat.

Dengan cara ini maka untuk melakukan proses loading untuk masing-masing sprite cukup menggunakan BLOCKREAD pada file PIC yang berisi semua sprite Anda dengan ukuran yang dapat Anda lihat pada file berekstensi SZS yang berisi informasi dari tiap sprite.

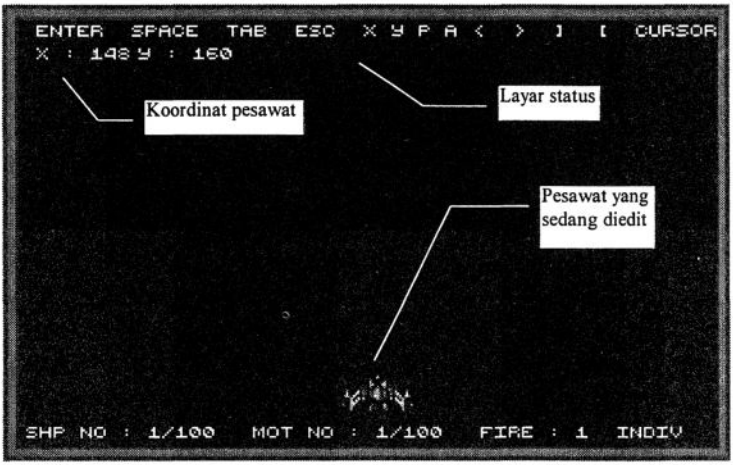
Hal yang paling penting dalam pembuatan editor adalah efisiensi dan kemudahan pemakaian. Karena dua hal ini dapat mempengaruhi game yang Anda buat. Bila editor yang Anda buat sangat rumit dan sangat sulit mengoperasikannya, orang yang bertugas merancang level akan menjadi malas untuk memakainya.

PEMAKAIAN FIRE FIGHTER MISSION EDITOR

Ini adalah tampilan yang akan Anda dapat pada saat pertama kali Anda menjalankan Editor. Untuk memilih gunakan tombol kursor dan untuk eksekusi gunakan Enter.

EDIT SHIP

Di sini Anda dapat mengatur route dari pesawat musuh. Di mana posisinya pada setiap frame, kapan dia melakukan penembakan, berapa besar powernya dan sebagainya. Tampilan yang akan Anda dapat adalah sebagai berikut.



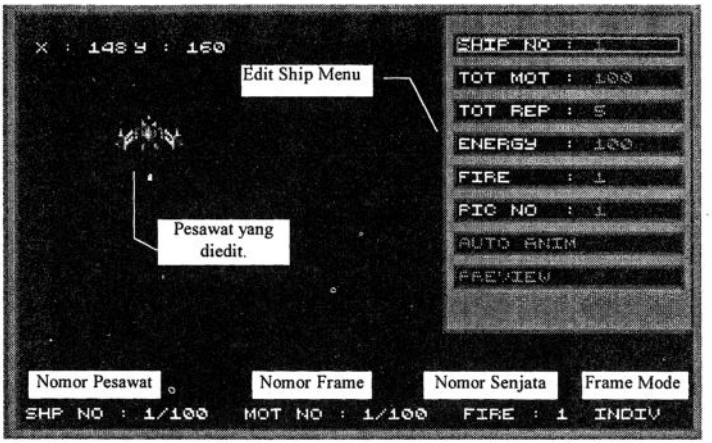
Layar Status:

- Tombol yang dapat Anda tekan ditampilkan pada baris pertama, gunanya untuk menunjukkan tombol yang dapat Anda tekan pada saat itu.
- Koordinat X dan Y dari pesawat pada baris kedua dari atas (X: 148 Y: 160).
- Nomor pesawat yang sedang disunting (SHP NO: 1/100).
- Nomor frame yang sedang aktif (MOT NO: 1/100).
- Senjata yang ditembakkan (FIRE: 1).
- Mode dari pemindahan posisi pesawat (INDIV).

Tombol Fungsi :

- Tekan Enter untuk mengaktifkan/menon-aktifkan mode pemindahan pesawat. Bila mode pemindahan pesawat aktif, maka pesawat akan bergerak mengikuti tombol kursor yang Anda tekan dan pesawat akan dikelilingi oleh persegi panjang putih.

- Tekan Space Bar untuk mengaktifkan/menonaktifkan mode follow. Bila mode follow nonaktif maka akan ditampilkan tulisan INDIV yang berarti individual (seperti pada display di atas). Jika Anda melakukan pemindahan pesawat dengan mode follow aktif, maka Anda juga menggerakkan posisi pesawat pada frame-frame berikutnya (Mulai frame itu sendiri hingga frame terakhir). Sistem ini diciptakan untuk memberikan kemudahan dalam mengedit posisi pesawat. Pada saat Anda menciptakan pesawat baru, posisi pesawat pada setiap frame otomatis diletakkan pada koordinat (0,0). Untuk membuat pesawat tersebut berjalan mengikuti route, Anda harus memindahkan posisi pesawat tersebut pada frame-frame berikutnya. Tentu saja proses pemindahan pesawat ini harus Anda lakukan satu-persatu setiap framenya. Bayangkan jika mode follow tidak ada, Anda terpaksa harus menggerakkan pesawat tersebut mulai koordinat (0,0) lagi hingga posisi yang benar setiap framenya. Dengan mode follow ini, maka Anda cukup memindahkan pesawat tersebut mulai dari koordinat pesawat pada frame sebelumnya ke posisi yang baru, karena pada saat Anda menggerakkan pesawat pada frame sebelumnya, posisi pesawat pada frame-frame berikutnya ikut bergeser (tentu saja lebih dekat dari pada menggerakkannya mulai dari koordinat (0,0) lagi). Agar Anda mendapat gambaran, cobalah proses pemindahan pesawat dengan mode follow dan mode individual. Lihat perbedaannya.
- Tekan Tab untuk mengaktifkan/menon-aktifkan data menu. Di sini Anda dapat mengisikan data dari pesawat seperti jumlah power, nomor senjata yang ditembakkan dan lain sebagainya. Untuk memilih data yang hendak Anda ubah gunakan kursor atas dan bawah kemudian tekan Enter.



Isi dengan data yang baru setelah itu tekan Enter lagi.

- SHIP NO adalah nomor pesawat yang sedang Anda sunting. Pesawat nomor 1 adalah pesawat Anda sendiri, sisanya adalah pesawat musuh.
- TOT MOT adalah jumlah frame yang dipakai oleh pesawat tersebut.
- TOT REP adalah jumlah sprite gambar yang hendak diulang. Pada pesawat di atas, jumlah sprite gambar yang diulang adalah 5. Editor secara otomatis akan mengulang gambar sprite mulai nomor yang Anda isikan pada (PIC NO) hingga (PIC NO + 4), untuk itu pada saat Anda melakukan proses capture, sebaiknya Anda melakukan capture pada sprite-sprite yang akan diulang secara berurutan sehingga memudahkan proses edit.
- ENERGY adalah jumlah power yang dimiliki oleh pesawat tersebut.
- FIRE menunjukkan nomor senjata yang ditembakkan oleh pesawat pada frame itu. Jangan mengisikan nomor senjata yang belum Anda buat di EDIT WEAPON.
- PIC NO menunjukkan nomor sprite yang dipakai pada frame itu. Dengan fasilitas ini Anda dapat melakukan variasi sprite pada animasi pesawat, jadi sprite yang akan ditampilkan tidak selalu perulangan sprite mulai nomor (PIC NO) hingga (PICNO + TOT REP) bahkan kalau Anda mau, Anda bisa membuat pesawat tersebut berubah menjadi robot atau menjadi pesawat lain dengan

cara mengisikan sprite-sprite perubahan pesawat tersebut pada frame-frame secara berurutan.

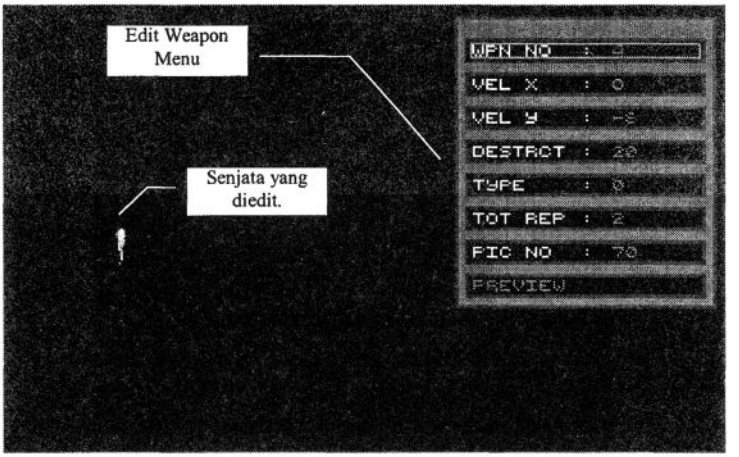
- AUTO ANIM digunakan untuk mengisi PIC NO pada semua frame dengan perulangan sprite mulai nomor PIC NO hingga (PIC NO + TOT REP). Contohnya bila TOT REP = 3, PIC NO = 10 kemudian Anda Anda lakukan AUTO ANIM. PIC NO pada frame 1 akan diisi dengan 10, pada frame 2 berisi 11, pada frame 3 berisi 12, pada frame 4 kembali ke 10 pada frame 5 berisi 11, demikian seterusnya hingga frame terakhir. Bayangkan bila fasilitas ini tidak ada, Anda harus mengisi PIC NO satu persatu pada setiap frame.
- PREVIEW berguna untuk menjalankan pesawat tersebut mengikuti routenya sehingga Anda dapat melihat secara langsung bagaimana model route pesawat tersebut.
- ESC berguna untuk keluar ke menu utama.
- Tekan X untuk menyalin perubahan koordinat X pesawat antara frame sebelumnya dan frame yang sekarang ke frame-frame selanjutnya. Proses ini dibuat sebagai shortcut, bila Anda ingin menggerakkan pesawat tersebut dua pixel ke kanan pada setiap framenya, walaupun sudah diberikan fasilitas follow dimana Anda tinggal menggerakkan pesawat tersebut dua pixel ke kanan tetapi hal ini masih juga membuang waktu. Cara ini dapat Anda persingkat sebagai berikut. Masuklah ke frame 2, pindahkan pesawat dua pixel ke kanan. Tekan X kemudian tekan Y untuk memberikan jawaban YES. Secara otomatis mulai frame 3 hingga frame yang terakhir, pesawat akan bergeser ke kanan dua pixel. Fasilitas ini tidak dapat Anda gunakan pada frame 1 karena pada frame 1 posisi pesawat belum berubah. Pada saat Anda melakukan proses ini, mode pemindahan pesawat harus aktif.
- Tekan Y untuk menyalin perubahan koordinat Y pesawat pada frame sebelumnya ke frame yang sekarang ke frame-frame berikutnya. Prosesnya sama persis dengan penekanan tombol X di atas.
- Tekan P untuk meletakkan pesawat pada posisi pesawat di frame sebelumnya. Sekali lagi, mode pemindahan pesawat harus aktif.
- Tekan A untuk melakukan AUTO USER ANIM. Mode pemindahan pesawat harus aktif. Dengan memilih fasilitas ini, Anda dapat melakukan pengulangan sprite yang dipakai pada frame 1 hingga frame sekarang untuk diisikan pada frame-frame berikutnya.

Misalnya sebagai berikut. Anda sedang berada pada frame 4. PIC NO pada frame 1 berisi 10, PIC NO pada frame 2 berisi 13, PIC NO pada frame 3 berisi 14. Tekan tombol Enter untuk mengaktifkan mode pemindahan pesawat, kemudian tekan A dan jawab yes (Y). PIC NO pada frame 4 akan berisi 10, pada frame 5 akan berisi 13, pada frame 6 akan berisi 14, pada frame 7 akan berisi 10 lagi, demikian seterusnya hingga frame terakhir.

- Tekan < untuk masuk ke frame sebelumnya.
- Tekan > untuk masuk ke frame berikutnya.
- Tekan [untuk mengedit pesawat sebelumnya.
- Tekan] untuk mengedit pesawat berikutnya.

EDIT WEAPON

Pada edit weapon, Anda dapat menciptakan bermacam-macam senjata dengan berbagai karakteristiknya, seperti daya rusaknya, kecepatannya, tipenya dan sebagainya. Pada saat memasuki edit weapon, akan terlihat tampilan sebagai berikut.



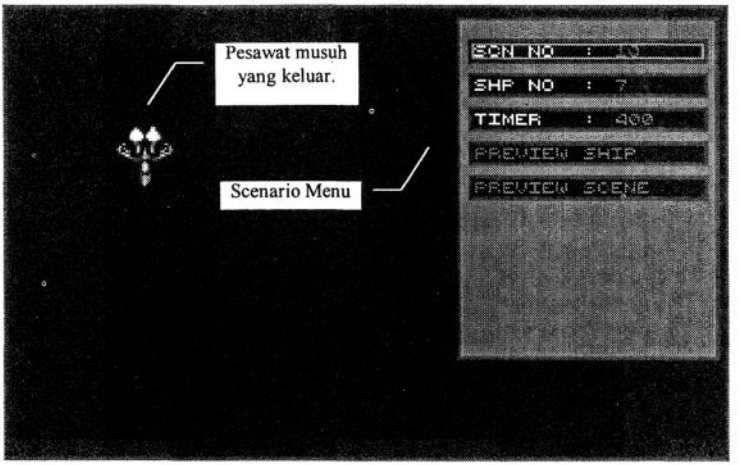
Tombol Fungsi:

- Tekan kursor atas dan bawah untuk melakukan pemilihan data yang hendak Anda ubah.
- Tekan Enter untuk eksekusi.

- WPN NO adalah nomor senjata yang Anda edit. Nomor 1 sampai nomor 4 adalah senjata dari pesawat kita. Senjata nomor 5 adalah bomb pesawat kita sedangkan sisanya dapat Anda gunakan untuk senjata pesawat musuh.
- VEL X adalah kecepatan sumbu X dari senjata tersebut (lompatan per pixel pada satu loop).
- VEL Y adalah kecepatan sumbu Y dari senjata tersebut (lompatan per pixel pada satu loop).
- DESTRCT adalah daya rusak dari senjata tersebut.
- TYPE adalah tipe dari senjata tersebut.
  - TYPE 0 atau TYPE 1 adalah senjata biasa yang mempunyai kecepatan dan arah yang sudah pasti.
  - TYPE 3 adalah senjata yang arahnya ditentukan pada saat melakukan penembakan yaitu ke arah pesawat kita. Kecepatan jalannya peluru ditentukan dengan besarnya VEL X.
  - TYPE 5 adalah senjata yang perubahan sumbu X-nya mengikuti arah pesawat kita sehingga senjata tersebut terlihat seperti semi-mengejar.
- TOT REP adalah jumlah perulangan sprite. Sama dengan TOT REP pada EDIT SHIP.
- PIC NO adalah nomor sprite yang digunakan. Sama dengan PIC NO pada EDIT SHIP.
- PREVIEW digunakan untuk melihat route senjata secara langsung. Untuk kembali ke layar penyuntingan, tekan tombol ESC.
- ESC untuk kembali ke menu utama.

EDIT SCENARIO

Berguna untuk merancang level dari permainan, menentukan kapan sebuah pesawat musuh harus dikeluarkan. Ini adalah bagian yang terpenting dari editor. Tampilan menu dari EDIT SCENARIO adalah sebagai berikut.



Tombol Fungsi :

- Tekan kursor atas dan bawah untuk memilih data yang hendak diganti. Kemudian tekan Enter untuk eksekusi.
  - SCN NO adalah nomor frame skenario yang sedang disunting.
  - SHP NO adalah nomor pesawat yang muncul pada nomor scenario ini. Jangan mengosongkan sebuah frame skenario bila frame skenario itu bukan yang terakhir karena game engine akan menganggap scenario sudah selesai apabila ditemukan frame scenario yang SHP NO-nya kosong. Dan jangan sekali-kali memasukkan nomor pesawat yang belum Anda buat sama sekali. Satu hal lagi, karena buffer dari pesawat yang dapat dikeluarkan berjumlah 20, maka usahakan tidak terjadi suatu situasi dimana jumlah pesawat yang aktif lebih dari 20. Apabila hal ini terjadi maka pesawat yang ke 21 dan seterusnya tidak akan dikeluarkan hingga ada pesawat yang non active. Untuk itu dalam merancang skenario, jika telah ada 20 pesawat yang sedang aktif, perhitungkan pada loop ke berapa ada pesawat yang frame-nya (TOT MOT) sudah habis sehingga oleh game engine dinonaktifkan dari buffer. Nah pada saat itu Anda dapat menambahkan sebuah pesawat lagi.
  - TIMER adalah waktu di mana pesawat tersebut harus dikeluarkan. Ingat, dalam mengisi TIMER harus secara berurutan. Frame Scenario yang lebih awal harus memiliki

TIMER paling tidak sama atau lebih kecil dari frame-frame berikutnya karena frame scenario ini nantinya akan dijalankan secara berurutan. Satu TIMER sama dengan satu looping pada game engine.

- PREVIEW SHIP digunakan untuk melihat route dari pesawat yang Anda pilih (SHP NO).
- PREVIEW SCENE digunakan untuk menjalankan skenario secara keseluruhan. Dengan ini Anda dapat melihat hasil dari level yang Anda rancang tanpa harus menjalankannya melalui game engine. Animasi yang dijalankan benar-benar sudah seperti animasi yang dihasilkan oleh game engine. Tekan ESC untuk kembali ke layar penyuntingan.

Gambar 4.1 Tampilan dari Perview Scene.

Gambar 4.2 Tampilan dari Perview Scene.

Tekan ESC untuk kembali ke menu utama.

SAVE MISSION

Berguna untuk menyimpan level yang telah Anda rancang. Di sini semua data seperti data pesawat, scenario dan lain-lainnya akan disimpan ke dalam file.

QUIT

Berguna untuk keluar dari editor.

Sekarang untuk beberapa saat bermain-mainlah dulu dengan FIRE FIGHTER MISSION EDITOR untuk lebih memberikan gambaran mengenai pentingnya sebuah editor bagi Anda. Cobalah untuk mengganti beberapa route pesawat, senjata yang ditembakkannya, bahkan cobalah mengganti gambarnya. Setelah itu Anda dapat mencoba mengedit skenario untuk menambah ataupun mengurangi pesawat yang akan muncul. Tapi ingat perubahan itu jangan Anda simpan terlebih dahulu karena akan mengubah data yang lama.

PEMBUATAN FIRE FIGHTER MISSION EDITOR

Semua fasilitas dan teknik yang digunakan dalam pembuatan editor ini telah kita bahas pada bab-bab sebelumnya, karena itu sangatlah penting bahwa Anda telah benar-benar menguasainya.

Seperti biasa, lihat dan pelajarilah listing program FIRE FIGHTER MISSION EDITOR di bawah ini. Mungkin Anda sudah bisa mempelajarinya tanpa perlu membaca penjelasan program. Ada baiknya mencoba mempelajari program secara langsung.

Listing Program

```
{-------------------------------------------------------------}
{ File       :  Editor.Pas                                    }
{ Compiler   :  Turbo Pascal 7.00                             }
{ Fungsi     :  Editor dari game Fire Fighter.                }
{-------------------------------------------------------------}

Uses Crt,VgaCad,VgaFont,Dos;

Const Frame_Per_Sec = 35;

Type
    Rec_Motion = Record
      Xp,Yp : Integer;
      Fire  : Byte;
      Pic_No: Byte;
    End;

    Rec_Ship = Record
      Tot_Mot: Byte;
      Tot_Rep: Byte;
      Motion: Array[1..200] of Rec_Motion;
      Energy: Byte;
    End;

    Rec_Ship_Status = Record
      Shp_No: Byte;
      Ctr   : Integer;
      Active: Boolean;
      Energy: Byte;
    End;
```

```
    Rec_Weapon = Record
      Vx,Vy    : Integer;
      Pic_No   : Byte;
      Destruct: Byte;
      Tipe     : Byte;
      Tot_Rep  : Byte;
    End;

    Rec_Weapon_Status = Record
      Xp,Yp : Integer;
      Wpn_No: Byte;
      Ctr   : Byte;
    End;

    Cell_Frame = Record
      Pic_No: Byte;
    End;

    Rec_Anim = Record
      Frame : Array[1..20] of Cell_Frame;
      Total : Integer;
    End;

    Rec_Anim_Status = Record
      Xp,Yp : Integer;
      Ctr   : Integer;
      Active: Boolean;
    End;

    Rec_Sprite = Record
      Length, Width: Integer;
      Picture      : Pointer;
    End;

    Var_Ed = Record
      Shp_No, Mot_Ctr: Byte;
      Wpn_No, Frm_Ctr: Byte;
      Tag,Poss_Mode  : Boolean;
      Scene_No       : Integer;
      Time           : Longint;
    End;

    Rec_Scene = Record
      Shp_No: Byte;
      Timer : Integer;
    End;

Var Scenario     : Array[1..5000] of Rec_Scene;
    Bank_Ship    : Array[1..100]  of ^Rec_Ship;
```

```
    Ship          : Array[1..20]   of Rec_Ship_Status;
    Bank_Weapon   : Array[1..20]   of Rec_Weapon;
    Weapon        : Array[1..100]  of Rec_Weapon_Status;
    Picture       : Array[1..255]  of Rec_Sprite;
    Buff          : Pointer;
    Loop          : Integer;
    Loop_Time     : Integer;
    Proc          : Pointer;
    X,Y           : Integer;
    Luar          : Boolean;
    Scroll        : Integer;
    Option        : Byte;
    Ch            : Char;
    File_Data     : File;
    Editor        : Var_Ed;

Procedure Small;External;
{$L Small.Obj}

Function Input(X,Y,L,Code:Integer): String;
Var Luar:Boolean;
    Nam:String[100];
  Begin
    B:=0;
    Nam:='';
    Luar:=False;
    Repeat
      Repeat
        SetChar56Colors(244,244,244,244,244,244);
        If B<L then PutString56(Screen^,X,Y,Nam+'-') else
        PutString56(Screen^,X,Y,Nam);
      Until Keypressed;
      Ch:=Readkey;
      SetChar56Colors(222,222,222,222,222,222);
      PutString56(Screen^,X,Y,Nam+'-');
      If (Ch=#8) and (B>0) then
        Begin
          SetChar56Colors(222,222,222,222,222,222);
          PutString56(Screen^,X,Y,Nam);
          Delete(Nam,B,1);
          Dec(B);
        End Else
      Begin
       If (Ch in ['0'..'9','a'..'z','A'..'Z']) and (Code=1) and (B<L) then
          Begin
            Inc(B);
            Nam:=Nam+Ch;
          End;
       If (Ch in ['0'..'9','a'..'z','A'..'Z',' ']) and (Code=3) and (B<L) then
          Begin
```

```
            Inc(B);
            Nam:=Nam+Ch;
          End;
        If (Ch in ['-','0'..'9']) and (Code=0) and (B<L) then
          Begin
            Inc(B);
            Nam:=Nam+Ch;
          End;
      End;
      If Ch=#13 then Luar:=True;
      If Ch=#27 then
        Begin
          Luar:=True;
          Nam:=' ';
        End;
    Until Luar;
    Input:=Nam;
    SetChar56Colors(244,244,244,244,244,244);
    PutString56(Screen^,X,Y,Nam);
  End;

Procedure Beep;
  Begin
    Sound(70);Delay(10);Nosound;
  End;

Function Strig(Nb:Integer): String;
Var S:String[10];
  Begin
    Str(Nb,S);
    Strig:=S;
  End;

Function Sinput(x,y,l,code,min,max,last_val:integer): Integer;
Var S:Integer;
    Wd:Word;
  Begin
    Block(Screen^,x,y,x+l*10,y+6,0);
    Val(Input(x,y,l,code),S,Wd);
    If (s>=min) and (s<=max) then Sinput:=S else
      Begin
        Sinput:=last_val;
        Sound(800);Delay(15);Nosound;
      End;
    Block(Screen^,x,y,x+l*10,y+6,0);
  End;

Procedure Hollow(Var Picbuff; X1,Y1,X2,Y2: Integer);
  Begin
    Block(PicBuff,X1,Y1,X2,Y2,0);
```

```
    Rectangle(Picbuff,X1,Y1,X2,Y2,210);
    Rectangle(Picbuff,X1+1,Y1+1,X2-1,Y2-1,214);
  End;

Procedure Windows(Var Picbuff; X1,Y1,X2,Y2: Integer);
  Begin
    Block(PicBuff,X1,Y1,X2,Y2,205);
    Rectangle(Picbuff,X1,Y1,X2,Y2,214);
    Rectangle(Picbuff,X1+1,Y1+1,X2-1,Y2-1,210);
  End;

Function Yes_No(St:String): Boolean;
Var Ch:Char;
  Begin
    GetImage(Screen^,10,150,180,165,Buff^);
    Windows(Screen^,10,150,180,165);
    PutString(Screen^,15,155,0,15,St+' (Y/N)?');
    Repeat
      Ch:=Readkey;
      Beep;
    Until Ch in ['y','Y','n','N'];
    If Ch in ['y','Y'] then Yes_No:=True else
      Yes_No:=False;
    PutImage(Screen^,10,150,Buff^);
  End;

Procedure Load_Data;
  Begin
    For X:=1 to 100 do
      GetMem(Bank_Ship[X],Sizeof(Rec_Ship));
    Assign(File_Data,'Ship.Dat');
    Reset(File_Data,1);
    BlockRead(File_Data,Bank_Ship[1]^,50*Sizeof(Rec_Ship));
    BlockRead(File_Data,Bank_Ship[51]^,50*Sizeof(Rec_Ship));
    Close(File_Data);

    Assign(File_Data,'Weapon.Dat');
    Reset(File_Data,1);
    BlockRead(File_Data,Bank_Weapon[1],Sizeof(Bank_Weapon));
    Close(File_Data);

    Assign(File_Data,'Scenario.Dat');
    Reset(File_Data,1);
    BlockRead(File_Data,Scenario,SizeOf(Scenario));
    Close(File_Data);
  End;

Procedure Load_Picture;
Var File_Size: Text;
    Szs      : Integer;
```

```
  Begin
    Assign(File_Data,'Pic\Picture.Pic');
    Reset(File_Data,1);
    Assign(File_Size,'Pic\Picture.Szs');
    Reset(File_Size);
    X:=0;
    Repeat
      Inc(X);
      Readln(File_Size,Szs);
      Readln(File_Size,Picture[X].Length);
      Readln(File_Size,Picture[X].Width);
      GetMem(Picture[X].Picture,Szs);
      BlockRead(File_Data,Picture[X].Picture^,Szs);
    Until (Eof(File_Size));
    Close(File_Size);
    Close(File_Data);
    Load_Scroll_Page('Back.Pic');
  End;

Procedure Global_Var;

  Begin
    Luar:=False;
    Option:=1;
    Loop:=1;
    With Editor do
      Begin
        Shp_No:=1;
        Mot_Ctr:=1;
        Wpn_No:=1;
        Frm_Ctr:=1;
        Tag:=False;
        Poss_Mode:=False;
        Scene_No:=1;
      End;
    GetMem(Buff,10000);
    GetMem(Proc,64000);
  End;

Procedure Initialize;

  Begin
    OpenVga256;
    Install_Palette('Pic\Picture.Pal');
    Load_Data;
    Load_Picture;
    LoadFont(@Small);
    Global_Var;
  End;
```

```
Procedure Ship_Background;
  Begin
    Windows(Screen^,0,0,319,199);
    Hollow(Screen^,5,5,315,195);
  End;

Procedure Menu_Background;
  Begin
    Windows(Screen^,200,5,315,150);
    Hollow(Screen^,205,15,310,25);
    Hollow(Screen^,205,30,310,40);
    Hollow(Screen^,205,45,310,55);
    Hollow(Screen^,205,60,310,70);
    Hollow(Screen^,205,75,310,85);
    Hollow(Screen^,205,90,310,100);
    Hollow(Screen^,205,105,310,115);
    Hollow(Screen^,205,120,310,130);
    PutString(Screen^,208,18,0,15,'SHIP NO: ');
    PutString(Screen^,208,33,0,15,'TOT MOT: ');
    PutString(Screen^,208,48,0,15,'TOT REP: ');
    PutString(Screen^,208,63,0,15,'ENERGY : ');
    PutString(Screen^,208,78,0,14,'FIRE   : ');
    PutString(Screen^,208,93,0,14,'PIC NO : ');
    PutString(Screen^,208,108,0,LightRed,'AUTO ANIM ');
    PutString(Screen^,208,123,0,LightRed,'PREVIEW');
  End;

Procedure Display_Edit_Menu_Data;
  Begin
    With Editor do
      With Bank_Ship[Shp_No]^ do
        Begin
          PutString(Screen^,270,18,0,LightRed,Strig(Shp_No));
          PutString(Screen^,270,33,0,LightRed,Strig(Tot_Mot));
          PutString(Screen^,270,48,0,LightRed,Strig(Tot_Rep));
          PutString(Screen^,270,63,0,LightRed,Strig(Energy));

PutString(Screen^,270,78,0,LightRed,Strig(Motion[Mot_Ctr].Fire));
PutString(Screen^,270,93,0,LightRed,Strig(Motion[Mot_Ctr].Pic_No));
        End;
  End;

Procedure Edit_Ship_Status(Tipe:Byte);
  Begin
    If Tipe = 1 then
      Begin
        Block(Screen^,6,6,314,194,0);
PutString(Screen^,15,9,0,15,'ENTER SPACE  TAB  ESC  X Y P A < > ]  [  CURSOR');
      End Else
      Block(Screen^,6,6,199,180,0);
```

```
PutString(Screen^,15,19,0,14,'X:'+String(Bank_Ship[Editor.Shp_No]^.Motion[Editor.Mot_Ctr].Xp));
PutString(Screen^,60,19,0,14,'Y:'+String(Bank_Ship[Editor.Shp_No]^.Motion[Editor.Mot_Ctr].Yp));
    With Editor do
      Begin
        With Bank_Ship[Shp_No]^ do
          Begin
            If (Motion[Mot_Ctr].Pic_No<>0) then
              If Tipe = 1 then
                Begin
PutImage(Screen^,Motion[Mot_Ctr].Xp,Motion[Mot_Ctr].Yp,Picture[[Mot_Ctr].Pic_No].Picture^);
                  If Tag then

Rectangle(Screen^,Motion[Mot_Ctr].Xp,Motion[Mot_Ctr].Yp,

Motion[Mot_Ctr].Xp+Picture[Motion[Mot_Ctr].Pic_No].Length,

Motion[Mot_Ctr].Yp+Picture[Motion[Mot_Ctr].Pic_No].Width,15);
                  If Motion[Mot_Ctr].Fire<>0 then
                    Begin
If Bank_Weapon[Motion[Mot_Ctr].Fire].Pic_No<>0 then

PutImage(Screen^,Motion[Mot_Ctr].Xp+Picture[Motion[Mot_Ctr].Pic_No].Length Div 2,
Motion[Mot_Ctr].Yp+Picture[Motion[Mot_Ctr].Pic_No].Width,

Picture[Bank_Weapon[Motion[Mot_Ctr].Fire].Pic_No].Picture^);
                    End;
                  End Else
                  Begin

PutImage(Screen^,50,50,Picture[Motion[Mot_Ctr].Pic_No].Picture^);
                  If Motion[Mot_Ctr].Fire<>0 then
                    Begin
If Bank_Weapon[Motion[Mot_Ctr].Fire].Pic_No<>0 then
PutImage(Screen^,50+Picture[Motion[Mot_Ctr].Pic_No].Length Div 2,
50+Picture[Motion[Mot_Ctr].Pic_No].Width,

Picture[Bank_Weapon[Motion[Mot_Ctr].Fire].Pic_No].Picture^);
                    End;
                  End;
          End;
        Block(Screen^,10,185,310,194,0);
PutString(Screen^,10,185,0,15,'SHP NO: '+Strig(Shp_No)+'/100');
PutString(Screen^,110,185,0,14,'MOT NO: '+Strig(Mot_Ctr)+'/'+
        Strig(Bank_Ship[Shp_No]^.Tot_Mot));
PutString(Screen^,210,185,0,14,'FIRE:'+
Strig(Bank_Ship[Shp_No]^.Motion[Mot_Ctr].Fire));
        If Editor.Poss_Mode then
          PutString(Screen^,270,185,0,14,'FOLLOW') else
          PutString(Screen^,270,185,0,14,'INDIV ');
      End;
```

```
    Rectangle(Screen^,5,5,315,195,210);
    Rectangle(Screen^,6,6,314,194,214);
    Rectangle(Screen^,0,0,319,199,214);
    Rectangle(Screen^,1,1,318,198,210);
    Rectangle(Screen^,2,2,317,197,205);
    Rectangle(Screen^,3,3,316,196,205);
    Rectangle(Screen^,4,4,315,195,205);
  End;

Procedure Auto_Arrange;
  Begin
    If Yes_No('ARRANGE PICTURE ') then
      With Editor do
        With Bank_Ship[Shp_No]^ do
          Begin
            For X:=2 to Tot_Mot do
              Begin
       Motion[X].Pic_No:=Motion[1].Pic_No+((X-1) Mod (Tot_Rep));
              End;
          End;
  End;

Procedure Preview_Ship(Shp_No:Byte);
  Begin
    Block(Screen^,0,0,319,199,0);
      With Bank_Ship[Shp_No]^ do
        Begin
          X:=1;
          While (Motion[X].Pic_No<>0) and (X<=Tot_Mot) do
            Begin
              PutImage(Screen^,Motion[X].Xp,Motion[X].Yp,
              Picture[Motion[X].Pic_No].Picture^);
              Delay(40);
            Block(Screen^,Motion[X].Xp,Motion[Y].Yp,Motion[X].Xp+
              Picture[Motion[X].Pic_No].Length,
              Motion[X].Yp+Picture[Motion[X].Pic_No].Width,0);
              Inc(X);
            End;
        End;
    Menu_Background;
  End;

Procedure Fill_Data(Option:Byte);
  Begin
    With Editor do
      With Bank_Ship[Shp_No]^ do
        Case Option of
          1:Begin
              Shp_No:=SInput(270,17,3,0,1,100,Shp_No);
              Editor.Mot_Ctr:=1;
```

```
            End;
          2:Tot_Mot:=SInput(270,32,3,0,1,200,Tot_Mot);
          3:Tot_Rep:=SInput(270,47,2,0,1,20,Tot_Rep);
          4:Energy:=SInput(270,62,3,0,0,255,Energy);

5:Motion[Mot_Ctr].Fire:=SInput(270,77,2,0,0,20,Motion[Mot_Ctr].Fire);

6:Motion[Mot_Ctr].Pic_No:=SInput(270,92,3,0,1,255,Motion[Mot_Ctr].Pic_No);
          7:Auto_Arrange;
          8:Preview_Ship(Shp_No);
        End;
    Menu_Background;
    Display_Edit_Menu_Data;
    Edit_Ship_Status(2);
  End;

Procedure Edit_Ship_Menu;
Var Luar  : Boolean;
    Option: Byte;
  Begin
    Option:=1;
    Luar:=False;
    Menu_Background;
    Display_Edit_Menu_Data;
    Edit_Ship_Status(2);
    Repeat
      Repeat
Rectangle(Screen^,207,17+(Option-1)*15,308,14+(Option)*15-6,7);
        Delay(5);
Rectangle(Screen^,207,17+(Option-1)*15,308,14+(Option)*15-6,0);
      Until Keypressed;
      Ch:=Readkey;
      Beep;
      Case Ch of
        #72  :Dec(Option);
        #75  :Begin
                Dec(Editor.Shp_No);
                If Editor.Shp_No>200 then Editor.Shp_No:=1;
                If Editor.Shp_No<1 then Editor.Shp_No:=200;
                Menu_Background;
                Display_Edit_Menu_Data;
              End;
        #77  :Begin
                Inc(Editor.Shp_No);
                If Editor.Shp_No>200 then Editor.Shp_No:=1;
                If Editor.Shp_No<1 then Editor.Shp_No:=200;
                Menu_Background;
                Display_Edit_Menu_Data;
              End;
        #80  :Inc(Option);
```

```
        #13  :Fill_Data(Option);
        #27,#9:Luar:=True;
      End;
      If Option<1 then Option:=8;
      If Option>8 then Option:=1;
      Edit_Ship_Status(2);
    Until Luar;
  End;

Procedure Edit_Ship;
Var Luar:Boolean;
    Xpo,Ypo:Integer;
    Ix,Iy:Integer;
    Xps,Yps:Integer;
  Begin
    Ship_Background;
    Edit_Ship_Status(1);
    Luar:=False;
    Repeat
      Repeat
      Until Keypressed;
      Ch:=Readkey;
      Beep;
      With Bank_Ship[Editor.Shp_No]^.Motion[Editor.Mot_Ctr] do
        Begin
          Xpo:=Xp;
          Ypo:=Yp;
          Case Ch of
            #9:Begin
                 Ship_Background;
                 Edit_Ship_Menu;
                 Ship_Background;
               End;
            #75:If Editor.Tag then Dec(Xp);
            #77:If Editor.Tag then Inc(Xp);
            #80:If Editor.Tag then Inc(Yp);
            #72:If Editor.Tag then Dec(Yp);
            '4':If Editor.Tag then Dec(Xp,10);
            '6':If Editor.Tag then Inc(Xp,10);
            '2':If Editor.Tag then Inc(Yp,10);
            '8':If Editor.Tag then Dec(Yp,10);
            'a','A':If Editor.Tag then
                      If Yes_No('PREVIOUS LOOP ANIM') then
                       Begin
                         Y:=1;
For X:=Editor.Mot_Ctr to Bank_Ship[Editor.Shp_No]^.Tot_Mot do
                           Begin

Bank_Ship[Editor.Shp_No]^.Motion[X].Pic_No:=
```

```
Bank_Ship[Editor.Shp_No]^.Motion[Y].Pic_No;
                              Inc(Y);
                              If Y=Editor.Mot_Ctr then Y:=1;
                            End;
                        End;
            'p','P':If Editor.Tag then
                        Begin
                          If Editor.Mot_Ctr<>1 then
                            Begin

Xp:=Bank_Ship[Editor.Shp_No]^.Motion[Editor.Mot_Ctr-1].Xp;

Yp:=Bank_Ship[Editor.Shp_No]^.Motion[Editor.Mot_Ctr-1].Yp;
                              End;
                        End;
            'y','Y':If (Editor.Tag) and (Editor.Mot_Ctr>1) then
                        Begin

Yps:=Bank_Ship[Editor.Shp_No]^.Motion[Editor.Mot_Ctr].Yp;
Iy:=Yps-Bank_Ship[Editor.Shp_No]^.Motion[Editor.Mot_Ctr-1].Yp;
If Yes_No('REPEAT POSSITION') then
For X:=Editor.Mot_Ctr+1 to Bank_Ship[Editor.Shp_No]^.Tot_Mot do
With Bank_Ship[Editor.Shp_No]^.Motion[X] do
                                  Begin
                                    Inc(Yps,Iy);
                                    Yp:=Yps;
                                  End;
                        End;
            'x','X':If (Editor.Tag) and (Editor.Mot_Ctr>1) then
                        Begin
Xps:=Bank_Ship[Editor.Shp_No]^.Motion[Editor.Mot_Ctr].Xp;
Ix:=Xps-Bank_Ship[Editor.Shp_No]^.Motion[Editor.Mot_Ctr-1].Xp;
If Yes_No('REPEAT POSSITION') then
For X:=Editor.Mot_Ctr+1 to Bank_Ship[Editor.Shp_No]^.Tot_Mot do
With Bank_Ship[Editor.Shp_No]^.Motion[X] do
                                  Begin
                                    Inc(Xps,Ix);
                                    Xp:=Xps;
                                  End;
                        End;
            ' ':Editor.Poss_Mode:=Not Editor.Poss_Mode;
            #13:Editor.Tag:=Not Editor.Tag;
            #27:If Yes_No('EXIT TO MAIN MENU') then Luar:=True;
            '>','.':Inc(Editor.Mot_Ctr);
            '<',',':Dec(Editor.Mot_Ctr);
            ']','}':Begin
                      Inc(Editor.Shp_No);
                      Editor.Mot_Ctr:=1;
                    End;
```

```
            '[','{':Begin
                      Dec(Editor.Shp_No);
                      Editor.Mot_Ctr:=1;
                    end;
          End;
        With Editor do
          Begin
If Mot_Ctr>Bank_Ship[Shp_No]^.Tot_Mot then Mot_Ctr:=1;
If Mot_Ctr<1 then Mot_Ctr:=Bank_Ship[Shp_No]^.Tot_Mot;
            If Shp_No>100 then Shp_No:=1;
            If Shp_No<1 then Shp_No:=100;
          End;
        If Editor.Poss_Mode then
For X:=Editor.Mot_Ctr+1 to Bank_Ship[Editor.Shp_No]^.Tot_Mot do
            Begin
              Inc(Bank_Ship[Editor.Shp_No]^.Motion[X].Xp,Xp-Xpo);
              Inc(Bank_Ship[editor.Shp_No]^.Motion[X].Yp,Yp-Ypo);
            End;
        End;
      Edit_Ship_Status(1);
    Until Luar;
  End;

Procedure Menu_Weapon_Background;
  Begin
    Block(Screen^,1,1,318,198,0);
    Windows(Screen^,200,5,315,135);
    Hollow(Screen^,205,15,310,25);
    Hollow(Screen^,205,30,310,40);
    Hollow(Screen^,205,45,310,55);
    Hollow(Screen^,205,60,310,70);
    Hollow(Screen^,205,75,310,85);
    Hollow(Screen^,205,90,310,100);
    Hollow(Screen^,205,105,310,115);
    Hollow(Screen^,205,120,310,130);
    PutString(Screen^,208,18,0,15,'WPN NO : ');
    PutString(Screen^,208,33,0,15,'VEL X  : ');
    PutString(Screen^,208,48,0,15,'VEL Y  : ');
    PutString(Screen^,208,63,0,15,'DESTRCT: ');
    PutString(Screen^,208,78,0,14,'TYPE   : ');
    PutString(Screen^,208,93,0,14,'TOT REP: ');
    PutString(Screen^,208,108,0,14,'PIC NO : ');
    PutString(Screen^,208,123,0,LightRed,'PREVIEW');
  End;

Procedure Display_Weapon_Menu_Data;
  Begin
    With Editor do
      With Bank_Weapon[Wpn_No] do
        Begin
```

```
          PutString(Screen^,270,18,0,LightRed,Strig(Wpn_No));
          PutString(Screen^,270,33,0,LightRed,Strig(Vx));
          PutString(Screen^,270,48,0,LightRed,Strig(Vy));
          PutString(Screen^,270,63,0,LightRed,Strig(Destruct));
          PutString(Screen^,270,78,0,LightRed,Strig(Tipe));
          PutString(Screen^,270,93,0,LightRed,Strig(Tot_Rep));
          PutString(Screen^,270,108,0,LightRed,Strig(Pic_No));
        End;
  End;

Procedure Preview_Weapon;
Var Xp,Yp:Integer;
  Begin
    Block(Screen^,1,1,318,198,0);
    Xp:=155; Yp:=0;
    Editor.Frm_Ctr:=0;
    With Editor do
      With Bank_Weapon[Wpn_No] do
        Repeat
          Inc(Frm_Ctr);
          If Frm_Ctr>=Tot_Rep then Frm_Ctr:=0;

PutImage(Screen^,Xp,Yp,Picture[Pic_No+Frm_Ctr].Picture^);
          Delay(50);
          Block(Screen^,Xp,Yp,Xp+Picture[Pic_No+Frm_Ctr].Length,
          Yp+Picture[Pic_No+Frm_Ctr].Width,0);
          Inc(Xp,Vx);
          Inc(Yp,Vy);
          Rectangle(Screen^,0,0,319,199,LightBlue);
Until (Xp<0) or (Xp+Picture[Pic_No+Frm_Ctr].Length>319) or (Yp<0) or
      (Yp+Picture[Pic_No+Frm_Ctr].Width>199) or (Tipe=2);
  End;

Procedure Fill_Weapon_Data(Option:Byte);
  Begin
    With Editor do
      With Bank_Weapon[Wpn_No] do
        Case Option of
          1:Wpn_No:=SInput(270,17,2,0,1,20,Wpn_No);
          2:Vx:=SInput(270,32,4,0,-200,200,Vx);
          3:Vy:=SInput(270,47,4,0,-200,200,Vy);
          4:Destruct:=SInput(270,62,3,0,0,255,Destruct);
          5:Tipe:=SInput(270,77,1,0,0,10,Tipe);
          6:Tot_Rep:=SInput(270,92,2,0,1,20,Tot_Rep);
          7:Pic_No:=SInput(270,107,3,0,1,255,Pic_No);
          8:Preview_Weapon;
        End;
    Menu_Weapon_Background;
    Display_Weapon_Menu_Data;
  End;
```

```
Procedure Edit_Weapon_Status;
  Begin
    With Editor do
      With Bank_Weapon[Wpn_No] do
        Begin
          Block(Screen^,50,100,150,198,0);
If Pic_No<>0 then
PutImage(Screen^,50,100,Picture[Pic_No].Picture^);
        End;
  End;

Procedure Edit_Weapon;
Var Luar  : Boolean;
    Option: Byte;
  Begin
    Option:=1;
    Luar:=False;
    Menu_Weapon_Background;
    Display_Weapon_Menu_Data;
    Edit_Weapon_Status;
    Repeat
      Repeat
Rectangle(Screen^,207,17+(Option-1)*15,308,14+(Option)*15-6,7);
        Delay(15);
Rectangle(Screen^,207,17+(Option-1)*15,308,14+(Option)*15-6,0);
      Until Keypressed;
      Ch:=Readkey;
      Beep;
      Case Ch of
        #72  :Dec(Option);
        #80  :Inc(Option);
        #13  :Fill_Weapon_Data(Option);
        #27  :If Yes_No('EXIT TO MAIN MENU') then Luar:=True;
      End;
      If Option<1 then Option:=8;
      If Option>8 then Option:=1;
      Edit_Weapon_Status;
    Until Luar;
  End;

Procedure Scenario_Background;
  Begin
    Block(Screen^,1,1,318,198,0);
    Rectangle(Screen^,0,0,319,199,209);
    Rectangle(Screen^,1,1,318,198,214);
    Windows(Screen^,200,5,315,155);
    Hollow(Screen^,205,15,310,25);
    Hollow(Screen^,205,30,310,40);
    Hollow(Screen^,205,45,310,55);
```

```
    Hollow(Screen^,205,60,310,70);
    Hollow(Screen^,205,75,310,85);
    PutString(Screen^,208,18,0,15,'SCN NO : ');
    PutString(Screen^,208,33,0,15,'SHP NO : ');
    PutString(Screen^,208,48,0,15,'TIMER  : ');
    PutString(Screen^,208,63,0,LightRed,'PREVIEW SHIP');
    PutString(Screen^,208,78,0,LightRed,'PREVIEW SCENE');
  End;

Procedure Put_Sprite;
  Begin
    For X:=1 to 20 do
      If Ship[X].Active then
        With Bank_Ship[Ship[X].Shp_No]^.Motion[Ship[X].Ctr] do
          Begin
            If Pic_No<>0 then PutImage(Proc^,Xp,Yp,Picture
            [Pic_No].Picture^);
          End;
    For X:=1 to 100 do
      With Weapon[X] do
        If Wpn_No<>0 then
          Begin
            If Bank_Weapon[Wpn_No].Pic_No<>0 then

PutImage(Proc^,Xp,Yp,Picture[Bank_Weapon[Wpn_No].Pic_No].Picture^);
          End;
  End;

Procedure Do_Scenario(Var No_Data:Boolean);
  Begin
        While  (Scenario[Editor.Scene_No].Timer=Editor.Time)  and
(Scenario[Editor.Scene_No].Timer<>0) do
      Begin
        With Scenario[Editor.Scene_No] do
          If Shp_No<>0 then
            Begin
              For X:=1 to 20 do
                With Ship[X] do
                  If Not Active then
                    Begin
                      Active:=True;

Ship[X].Shp_No:=Scenario[Editor.Scene_No].Shp_No;
                      Ctr:=1;
                      Energy:=Bank_Ship[Ship[X].Shp_No]^.Energy;
                      X:=20;
                    End;
              Inc(Editor.Scene_No);
            End;
      End;
```

```
    If Scenario[Editor.Scene_No].Shp_No=0 then No_Data:=True;
  End;

Procedure Do_Ship_Status;
  Begin
    For X:=1 to 20 do
      With Ship[X] do
        If Active then
          Begin
            Inc(Ctr);
            If Bank_Ship[Shp_No]^.Motion[Ctr].Fire<>0 then
              Begin
                For Y:=1 to 100 do
                  With Weapon[Y] do
                    If Wpn_No=0 then
                      Begin

Weapon[Y].Wpn_No:=Bank_Ship[Shp_No]^.Motion[Ship[X].Ctr].Fire;

Weapon[Y].Xp:=Bank_Ship[Shp_No]^.Motion[Ship[X].Ctr].Xp+(Picture[
Bank_Ship[Shp_No]^.
Motion[Ship[X].Ctr].Pic_No].Length Div 2);

Weapon[Y].Yp:=Bank_Ship[Shp_No]^.Motion[Ship[X].Ctr].Yp+Picture[Bank_Ship[Shp_No]^.
                        Motion[Ship[X].Ctr].Pic_No].Width;
                        Weapon[Y].Ctr:=1;
                        Y:=100;
                      End;
              End;
          End;
  End;

Procedure Do_Weapon_Status;
  Begin
    For X:=1 to 100 do
      With Weapon[X] do
        If Wpn_No<>0 then
          Begin
            Inc(Ctr);
            Inc(Xp,Bank_Weapon[Wpn_No].Vx);
            Inc(Yp,Bank_Weapon[wpn_No].Vy);
If (Xp+Picture[Bank_Weapon[Wpn_No].Pic_No].Length>319) or (Xp<0)
or  (Yp+Picture[Bank_Weapon[Wpn_No].Pic_No].Width>199)  or  (Yp<0)
then Wpn_No:=0;
          End;
  End;

Procedure Do_Status(Var No_Data:Boolean);
  Begin
    Do_Scenario(No_Data);
```

```
    Do_Ship_Status;
    Do_Weapon_Status;
    Inc(Editor.Time);
    Dec(Scroll);
    If Scroll<1 then Scroll:=599;
  End;

Procedure Do_Checking(Var Luar:Boolean;No_Data:Boolean);
Var Still_Work:Boolean;
  Begin
    Still_Work:=False;
    For X:=1 to 20 do
      With Ship[X] do
        If Active then
          Begin
            If Ctr>Bank_Ship[Shp_No]^.Tot_Mot then
              Begin
                Ctr:=1;
                Active:=False;
              End;
            Still_Work:=True;
          End;
     If No_Data and (Still_Work = False) then Luar:=True;
  End;

Procedure Refresh_Data;
  Begin
    Editor.Time:=0;
    Editor.Scene_No:=1;
    For X:=1 to 20 do
      FillChar(Ship[X],SizeOf(Rec_Ship_Status),0);
    For X:=1 to 100 do
      FillChar(Weapon[X],SizeOf(Rec_Weapon_Status),0);
  End;

Procedure Put_Status;
  Begin
PutString(Proc^,5,180,0,14,'SCENE : '+Strig(Editor.Scene_No)+'/'+'5000');
    PutString(Proc^,120,180,0,14,'TIMER: '+Strig(Editor.Time));
  End;

Procedure Preview_Scenario;
Var Luar,No_Data:Boolean;
  Begin
    Fill_Page(Proc^,0);
    Refresh_Data;
    Luar:=False;
    No_Data:=False;
    Scroll:=599;
    Repeat
```

```
      Repeat
        Inc(Loop);
        If Loop=Loop_Time then
          Begin
            Put_Scroll_Page(Proc^,Scroll);
            Do_Status(No_Data);
            Do_Checking(Luar,No_Data);
            Put_Sprite;
            Put_Status;
            MoveTo(Proc^,Screen^);
            Loop:=1;
          End;

      Until (Keypressed) or (Luar);
      If Not Luar then
        Begin
          Ch:=Readkey;
          Beep;
          Case Ch of
            ' ':Begin
                  Ch:=Readkey;
                End;
            #27:Luar:=True;
          End;
        End;
    Until Luar;
  End;


Procedure Display_Scenario_Menu_Data;
  Begin
    With Editor do
      With Scenario[Scene_No] do
        Begin
          PutString(Screen^,270,18,0,LightRed,Strig(Scene_No));
          PutString(Screen^,270,33,0,LightRed,Strig(Shp_No));
          PutString(Screen^,270,48,0,LightRed,Strig(Timer));
        End;
  End;


Procedure Fill_Scenario_Data(Option:Byte);
  Begin
    With Editor do
      Case Option of
1:Scene_No:=SInput(270,17,3,0,1,5000,Scene_No);
2:Scenario[Scene_No].Shp_No:=SInput(270,32,3,0,0,100,Scenario[Scene_No].Shp_No);
3:Scenario[Scene_No].Timer:=SInput(270,47,5,0,1,30000,Scenario[Scene_No].Timer);
4:Preview_Ship(Scenario[Scene_No].Shp_No);
5:Preview_Scenario;
```

```
      End;
    Scenario_Background;
    Display_Scenario_Menu_Data;
  End;

Procedure Scenario_Status;
  Begin
    With Editor do
      With Bank_Ship[Scenario[Scene_No].Shp_No]^.Motion[1] do
        Begin

Block(Screen^,50,50,50+Picture[Pic_No].Length,50+Picture[Pic_No].
Width,0);
If (Pic_No<>0) and (Scenario[Scene_No].Shp_No<>0) then
     PutImage(Screen^,50,50,Picture[Pic_No].Picture^);
        End;
  End;

Procedure Edit_Scenario;

Var Luar   : Boolean;
    Option: Byte;
  Begin
    Option:=1;
    Luar:=False;
    Scenario_Background;
    Scenario_Status;
    Display_Scenario_Menu_Data;
    Repeat
      Repeat
Rectangle(Screen^,207,17+(Option-1)*15,308,14+(Option)*15-6,7);
        Delay(15);
Rectangle(Screen^,207,17+(Option-1)*15,308,14+(Option)*15-6,0);
      Until Keypressed;
      Ch:=Readkey;
      Beep;
      Case Ch of
        #72   :Dec(Option);
        #80   :Inc(Option);
        #13   :Fill_Scenario_Data(Option);
        #27   :If Yes_No('EXIT TO MAIN MENU') then Luar:=True;
      End;
      If Option<1 then Option:=5;
      If Option>5 then Option:=1;
      Scenario_Background;
      Scenario_Status;
      Display_Scenario_Menu_Data;
    Until Luar;
  End;
```

```
Procedure Save_Mission;
  Begin
    Assign(File_Data,'Ship.Dat');
    Rewrite(File_Data,1);
    BlockWrite(File_Data,Bank_Ship[1]^,50*Sizeof(Rec_Ship));
    BlockWrite(File_Data,Bank_Ship[51]^,50*Sizeof(Rec_Ship));
    Close(File_Data);

    Assign(File_Data,'Weapon.Dat');
    Rewrite(File_Data,1);
    BlockWrite(File_Data,Bank_Weapon[1],Sizeof(Bank_Weapon));
    Close(File_Data);

    Assign(File_Data,'Scenario.Dat');
    Rewrite(File_Data,1);
    BlockWrite(File_Data,Scenario,SizeOf(Scenario));
    Close(File_Data);
  End;

Procedure Execute;
  Begin
    Case Option of
      1:Edit_Ship;
      2:Edit_Weapon;
      3:Edit_Scenario;
      4:If Yes_No('SAVE') then Save_Mission;
      5:If Yes_No('EXIT') then Luar:=True;
    End;
  End;

Procedure Main_BackGround;
  Begin
    Fill_Page(Screen^,LightBlue);
    Block(Screen^,80,40,270,165,Blue);
    Block(Screen^,9,7,319,33,Blue);
    Windows(Screen^,4,2,315,28);
    Hollow(Screen^,10,10,309,22);
    PutString(Screen^,60,13,0,15,'FIRE FIGHTER MISSION EDITOR');
    Block(Screen^,9,175,319,195,Blue);
    Windows(Screen^,4,170,315,190);
    Hollow(Screen^,10,175,309,185);
    PutString(Screen^,40,177,0,14,'PROGRAM & DESIGN BY AGUSTINUS NALWAN 1994');
    Windows(Screen^,70,36,260,160);
    Hollow(Screen^,90,50,240,65);
    Hollow(Screen^,90,70,240,85);
    Hollow(Screen^,90,90,240,105);
    Hollow(Screen^,90,110,240,125);
    Hollow(Screen^,90,130,240,145);
    PutString(Screen^,120,55,LightRed,14,'1. EDIT SHIP');
    PutString(Screen^,120,75,LightRed,14,'2. EDIT WEAPON');
```

```
    PutString(Screen^,120,95,LightRed,14,'3. EDIT SCENARIO');
    PutString(Screen^,120,115,LightRed,14,'4. SAVE MISSION');
    PutString(Screen^,120,135,LightRed,14,'5. QUIT');
  End;

Procedure Main;
  Begin
    Main_Background;
    Repeat
      Repeat
Rectangle(Screen^,110,52+(Option-1)*20,230,50+(Option)*20-8,15);
        Delay(15);
Rectangle(Screen^,110,52+(Option-1)*20,230,50+(Option)*20-8,0);
      Until Keypressed;
      Ch:=Readkey;
      Beep;
      Case Ch of
        #80:Inc(Option);
        #72:Dec(Option);
        #27:Begin
              If Yes_No('EXIT') then Luar:=True;
            End;
        #13:Begin
              Execute;
              Main_Background;
            End;
      End;
      If Option>5 then Option:=1;
      If Option<1 then Option:=5;
    Until Luar;
  End;

Procedure Check_Timer;
Var Sec,Hour,MSec,Min:Word;
    T_Ctr:Integer;
  Begin
    Clrscr;
    TextBackground(1);
    TextColor(15);
Writeln('                    FIGHTER 1.00                                ');
    Writeln;
    TextBackground(0);
    Textcolor(7);
    Writeln('CLOCK TICK TESTING ');
    Writeln(Frame_Per_Sec,' Frames Per Second On Request ');
    Writeln('--------------------------------');
    SetTime(0,0,0,0);
    For T_Ctr:=1 to 10000 do
    GetTime(Hour,Min,Sec,MSec);
    Loop_Time:=Round((10000/Frame_Per_Sec)/(Sec+(MSec/100)));
```

```
    Writeln;
    Writeln('Speed Has Already Detected ');
    Writeln('Total Ticks Per Frame = ',Loop_Time);
    Writeln;
    Delay(1000);
    Writeln('Detected Speed Trial');
    For X:=1 to Loop_Time do
      Begin
        Write('.');
        For Y:=1 to Loop_Time do Y:=Y;
      End;
    Sound(200);Delay(30);Nosound;
    Delay(1000);
    Writeln;
    Writeln('Press Enter To Continue');
    Ch:=Readkey;
  End;

Begin
  Check_Timer;
  Initialize;
  Main;
  CloseVga256;
End.
```

PENJELASAN LISTING PROGRAM.

Lihatlah bagian USES dari program di atas. Di sana terlihat bahwa unit buatan yang digunakan adalah VGACAD dan VGAFONT. VGAFONT adalah UNIT yang berfungsi untuk memanipulasi FONT sehingga kita dapat menampilkan tulisan pada mode graphic. Pada saat editor ini saya buat, saya baru menyadari bahwa unit ini diperlukan untuk menampilkan data pada layar monitor. Pembuatan unit ini cukup rumit dan tidak akan dibahas pada buku ini karena procedure-procedurenya kebanyakan dirakit dengan bahasa ASSEMBLER. Untuk itu kita cukup mengenal dan mempelajari bagaimana cara pemakaiannya. Ingat, untuk melanjutkan ke bagian pembahasan program di bawah ini, Anda harus sudah mencoba FIRE FIGHTER MISSION EDITOR, karena mempelajari sesuatu yang belum pernah Anda coba tentu saja jauh lebih sulit daripada sesuatu yang sudah Anda kenal.

STRUKTUR DATA

- Frame_Per_Sec = Konstanta yang menunjukkan jumlah frame yang ditampilkan tiap detik..
- Scenario = Variabel Array of Rec_Scene yang berisi:
  - Shp_No = Variabel byte yang berisi nomor pesawat yang akan muncul pada layar permainan.
  - Timer = Variabel integer yang berisi waktu di mana pesawat tersebut akan muncul di layar permainan.
- Bank_Ship = Variabel Array of ^Rec_Ship. Digunakan pointer dengan tujuan untuk menghemat memori yang dipergunakan untuk variabel karena sebagaima kita ketahui bahwa variabel yang dideklarasikan tidak boleh melebihi 64 kbytes. Dengan cara ini maka jumlah variabel memori yang dipergunakan cukup 4 bytes (variabel pointer butuh 4 bytes) per cell arraynya atau = 4 x 100 = 400 bytes. Bila kita memakai Array of Rec_Scene, jumlah memori yang dipergunakan adalah 1203 bytes setiap cell arraynya = 100 x 1203 = 12030 bytes.
  - Tot_Mot = Variabel byte yang berisi jumlah frame dari sebuah pesawat. Setiap terjadi pergantian frame, terjadi pula perpindahan posisi pesawat. Jadi bila Tot_Mot berharga 40, pesawat tersebut akan berada pada layar monitor selama 40 x pergantian frame.
  - Tot_Rep = Variabel byte yang berisi jumlah sprite yang diulang.
  - Motion = Variabel Array of Rec_Motion. Setiap cell arraynya menyimpan data sebuah frame dari pesawat.
    - Xp = Koordinat X dari pesawat pada frame itu.
    - Yp = Koordinat Y dari pesawat pada frame itu.
    - Fire = Nomor senjata yang ditembakkan pada frame itu.

- Pic_No = Nomor sprite yang dipakai pada frame itu.
- Energy = Variabel byte yang berisi jumlah power dari pesawat.

- Ship = Variabel Array of Rec_Ship_Status. Fungsi dari variabel ini adalah untuk menyimpan data dari pesawat yang sedang aktif di layar permainan.
  - Shp_No = Variabel byte yang menunjukkan nomor dari pesawat yang dipakai. Nomor ini berfungsi sebagai indeks pada Bank_Ship.
  - Ctr = Variable integer. Fungsinya adalah sebagai counter yang menunjukkan nomor frame yang sedang aktif. Harga dari variabel ini akan ditambah pada setiap dilakukan pergantian frame.
  - Active = Variabel boolean yang menunjukkan apakah pesawat tersebut aktif atau tidak.
  - Energy = Variabel byte yang menunjukkan jumlah power pesawat yang masih tersisa.
- Bank_Weapon = Variabel Array of Rec_Weapon. Berfungsi untuk menampung data dari semua senjata yang dipergunakan dalam permainan.
  - Vx,Vy = Variabel integer yang berisi kecepatan dari senjata.
  - Pic_No = Variabel byte yang menunjukkan nomor sprite yang dipakai.
  - Destruct = Variabel byte yang berisi daya rusak dari senjata.
  - Tipe = Variabel byte yang menunjukkan tipe dari senjata.
  - Tot_Rep = Variabel byte yang menunjukkan jumlah sprite yang diulang.
- Weapon = Variabel Array of Rec_Weapon_Status. Sama seperti variabel Ship, variabel ini

berfungsi untuk menyimpan senjata yang sedang aktif pada layar permainan.

- Xp,Yp = Variabel integer yang menyimpan posisi dari senjata pada layar permainan.
- Wpn_No = Variabel byte yang berisi nomor senjata yang dipakai. Wpn_No ini berfungsi sebagai indeks dari Bank_Weapon.
- Ctr = Variabel byte yang menyimpan nomor sprite yang dipergunakan. Pada setiap pergantian frame dilakukan penambahan nilai dari variabel ini.

- Picture = Variabel Array of Rec_Sprite. Di sini data dari tiap sprite disimpan.
  - Length = Panjang dari sprite.
  - Width = Lebar dari sprite.
- Picture = Variabel pointer yang menyimpan data dari sprite.
- Loop_Time = Variabel integer yang menunjukkan berapa kali loop harus dilakukan untuk setiap pergantian frame.
- Proc = Variabel pointer sebagai virtual page.
- Editor = Berisi data dari editor.
  - Shp_No = Nomor pesawat yang sedang disunting.
  - Mot_Ctr = Nomor frame pesawat yang sedang disunting.
  - Wpn_No = Nomor senjata yang sedang disunting.
  - Frm_Ctr = Nomor frame senjata yang sedang disunting.
  - Tag = Mode pemindahan pesawat active/non-active.
  - Poss_Mode = Follow mode aktif/non-active.
  - Scene_No = Nomor skenario yang sedang disunting.
  - Time = Jumlah pergantian frame yang telah dilakukan.

Dengan penjelasan struktur data ini paling tidak kita telah memiliki gambaran bagaimana program editor maupun game engine tersebut hendak kita buat.

PENJELASAN PROGRAM

Karena program yang kita buat ini adalah sebuah editor, maka pemrograman yang akan kita lakukan akan meliputi menu untuk mengisi data dan melakukan pilihan, juga meliputi animasi yang berfungsi sebagai preview/display dari yang kita sunting sehingga memudahkan proses penyuntingan.

Sekarang kita lihat dulu main program dari editor ini.

```
Begin
  Check_Timer;
  Initialize;
  Main;
  CloseVga256;
End
```

Pemanggilan procedure CHECK_TIMER dilakukan untuk mendeteksi kecepatan dari komputer yang Anda pergunakan. Tujuannya agar, animasi yang dihasilkan selalu memiliki kecepatan yang kurang lebih sama pada jenis-jenis komputer yang berlainan. Cobalah sekarang kita pelajari bagaimana prosesnya dilakukan.

```
Procedure Check_Timer;
Var Sec,Hour,MSec,Min:Word;
    T_Ctr:Integer;
  Begin
    Clrscr;
    TextBackground(1);
    TextColor(15);
    Writeln('              FIRE FIGHTER 1.00                    ');
    Writeln;
    TextBackground(0);
    Textcolor(7);
    Writeln('CLOCK TICK TESTING ');
    Writeln(Frame_Per_Sec,' Frames Per Second On Request ');
    Writeln('-------------------------------');
```

Set waktu ke 0.

```
    SetTime(0,0,0,0);
```

Dilakukan counter sebanyak 10000 dengan kecepatan CPU Anda.

```
For T_Ctr:=1 to 10000 do
```

Diambil waktu sekarang sehingga kita dapat mengetahui waktu yang diperlukan untuk melakukan counter sebanyak 10000 tadi.

```
GetTime(Hour,Min,Sec,MSec);
```

Jumlah loop pada main program yang harus dilakukan sebelum dilakukan pergantian frame dapat dicari dengan rumus :

Loop_Time = 10000/(Frame_Per_Detik x Waktu).

```
Loop_Time:=Round((10000/Frame_Per_Sec)/(Sec+(MSec/100)));
Writeln;
Writeln('Speed Has Already Detected ');
Writeln('Total Ticks Per Frame = ',Loop_Time);
Writeln;
Delay(1000);
```

Menampilkan titik sebanyak jumlah Loop_Time. Gunanya agar pemain dapat mengetahui berapa kecepatan CPU-nya.

```
Writeln('Detected Speed Trial');
For X:=1 to Loop_Time do
  Begin
    Write('.');
    For Y:=1 to Loop_Time do Y:=Y;
  End;
Sound(200);Delay(30);Nosound;
Delay(1000);
Writeln;
Writeln('Press Enter To Continue');
Ch:=Readkey;
End;
```

Seperti biasanya, hampir pada tiap program kita ada procedure INITIALIZE. Tentu Anda sudah tahu bahwa biasanya procedure ini kita gunakan untuk mengisi nilai awal dari variabel. Kita akan melihat proses apa yang dilakukan oleh procedure ini.

```
Procedure Initialize;
  Begin
```

Dilakukan instalasi VGA 320 x 200 256 colors.

```
OpenVga256;
```

Palette default adalah PICTURE.PAL yang dibaca dari file.

```
Install_Palette('Picture.Pal');
Load_Data;
Load_Picture;
```

LOADFONT adalah salah satu procedure dari VGAFONT.TPU, fungsinya adalah untuk meload font SMALL.OBJ.

```
  LoadFont(@Small);
  Global_Var;
End;
```

Procedure LOAD_DATA ini berisi proses-proses load data pesawat, senjata dan juga skenario.

```
Procedure Load_Data;
  Begin
```

Karena Bank_Ship adalah array of Pointer, maka kita perlu memesan sejumlah memori yang kita butuhkan untuk masing-masing cell array. Tentu saja besarnya adalah sebesar record Rec_Ship karena memori pada alamat pointer tersebut akan menampung data dari record Rec_Ship.

```
For X:=1 to 100 do
  GetMem(Bank_Ship[X],Sizeof(Rec_Ship));
```

File SHIP.DAT dibuka untuk diload ke dalam Bank_Ship. File ini memuat data pesawat musuh seperti posisi tiap frame, jumlah power, senjata yang ditembakkan pada tiap frame dan lain-lainya.

```
Assign(File_Data,'Ship.Dat');
Reset(File_Data,1);
```

Dilakukan dua kali pembacaan dengan BlockRead karena sekali melakukan BlockRead, maksimum data yang dapat dibaca adalah 64 kbytes.

```
    BlockRead(File_Data,Bank_Ship[1]^,50*Sizeof(Rec_Ship));
    BlockRead(File_Data,Bank_Ship[51]^,50*Sizeof(Rec_Ship));
    Close(File_Data);
```

File WEAPON.DAT dibuka untuk diload ke dalam Bank_Weapon. File ini memuat data dari setiap senjata seperti daya rusak, kecepatan, tipe dan lain-lainnya.

```
    Assign(File_Data,'Weapon.Dat');
    Reset(File_Data,1);
    BlockRead(File_Data,Bank_Weapon[1],Sizeof(Bank_Weapon));
    Close(File_Data);
```

File SCENARIO.DAT dibuka untuk diload ke dalam Scenario. File ini memuat skenario dari permainan seperti kapan sebuah pesawat musuh akan muncul pada layar monitor.

```
    Assign(File_Data,'Scenario.Dat');
    Reset(File_Data,1);
    BlockRead(File_Data,Scenario,SizeOf(Scenario));
    Close(File_Data);
  End;
```

Procedure LOAD_PICTURE ini berisi proses loading gambar yaitu gambar background yang akan discroll dan juga sprite-sprite.

```
Procedure Load_Picture;
Var File_Size: Text;
    Szs      : Integer;
  Begin
```

Buka file Picture.PIC yang berisi sprite-sprite dan juga file Picture.SZS yang berisi informasi mengenai panjang, lebar maupun size dari masing-masing sprite.

```
    Assign(File_Data,'Picture.Pic');
    Reset(File_Data,1);
    Assign(File_Size,'Picture.Szs');
    Reset(File_Size);
    X:=0;
```

Lakukan looping untuk membaca informasi dari sprite dari file Picture.SZS dan juga membaca sprite dari file PICTURE.PIC.

```
    Repeat
      Inc(X);
      Readln(File_Size,Szs);
      Readln(File_Size,Picture[X].Length);
      Readln(File_Size,Picture[X].Width);
      GetMem(Picture[X].Picture,Szs);
      BlockRead(File_Data,Picture[X].Picture^,Szs);
    Until (Eof(File_Size));
    Close(File_Size);
    Close(File_Data);
```

Load file BACK.PIC sebagai gambar background yang akan discroll.

```
    Load_Scroll_Page('Back.Pic');
  End;
```

Procedure GLOBAL_VAR berisi proses-proses pengisian nilai awal dari variabel.

```
Procedure Global_Var;
  Begin
    Luar:=False;
    Option:=1;
    Loop:=1;
```

Variabel Editor adalah record yang berisi informasi tentang frame yang sedang disunting. Sebagai contoh, apabila Anda sedang melakukan EDIT SHIP dan Anda sedang menyunting pesawat nomor 5 maka nilai Editor.Shp_No adalah 5.

```
    With Editor do
      Begin
        Shp_No:=1;
        Mot_Ctr:=1;
        Wpn_No:=1;
        Frm_Ctr:=1;
        Tag:=False;
        Poss_Mode:=False;
        Scene_No:=1;
      End;
```

Buff adalah variabel pointer yang berfungsi sebagai buffer. Kegunaanya akan segera kita bahas.

```
    GetMem(Buff,10000);
```

Proc adalah virtual page yang kita pergunakan.

```
    GetMem(Proc,64000);
  End;
```

Bagian Initialize telah kita ketahui. Sekarang kita akan menuju ke main program. Pada main program kita akan menampilkan menu dengan pilihan EDIT SHIP, EDIT WEAPON, EDIT SCENARIO, SAVE MISSION dan QUIT karena pada saat editor ini Anda jalankan, Anda diharuskan untuk melakukan pemilihan dari option-option di atas.

```
Procedure Main;
  Begin
```

Pemanggilan procedure Main_Background berguna untuk melakukan penggambaran tampilan dari menu utama editor.

```
    Main_Background;
    Repeat
      Repeat
```

Fungsi dipanggilnya procedure RECTANGLE di sini untuk menampilkan persegi panjang pada option yang sedang kita pilih. Pemanggilan RECTANGLE yang kedua berguna untuk menghapus persegi panjang tersebut. Dengan cara ini maka persegi panjang itu akan terlihat berkedip.

```
Rectangle(Screen^,110,52+(Option-1)*20,230,50+(Option)*20-8,15);
        Delay(15);
Rectangle(Screen^,110,52+(Option-1)*20,230,50+(Option)*20-8,0);
      Until Keypressed;
      Ch:=Readkey;
```

Procedure BEEP dipanggil untuk mengeluarkan suara.

```
      Beep;
      Case Ch of
        #80:Inc(Option);
        #72:Dec(Option);
        #27:Begin
              If Yes_No('EXIT') then Luar:=True;
            End;
        #13:Begin
```

Jika tombol Enter ditekan, panggil procedure Execute. Pada procedure Execute ini akan dilakukan deteksi option mana yang dipilih oleh user. Langkah selanjutnya adalah memanggil procedure yang berkenaan dengan option yang dipilih.

```
            Execute;
```

Dilakukan penggambaran background lagi. Proses ini dilakukan karena setelah kembali dari salah satu option misalnya dari EDIT SHIP, gambar di layar telah berubah total. Untuk itu menu utama perlu digambar lagi.

```
            Main_Background;
          End;
      End;
      If Option>5 then Option:=1;
      If Option<1 then Option:=5;
    Until Luar;
  End;
```

Procedure MAIN_BACKGROUND berfungsi untuk menggambar menu utama pada layar, Hampir semua fasilitas di bawah ini telah Anda kenali dengan baik karena telah dibahas pada bab-bab sebelumnya.

```
Procedure Main_BackGround;
  Begin
    Fill_Page(Screen^,LightBlue);
    Block(Screen^,80,40,270,165,Blue);
    Block(Screen^,9,7,319,33,Blue);
    Windows(Screen^,4,2,315,28);
    Hollow(Screen^,10,10,309,22);
```

Fasilitas ini belum Anda kenal. Procedure di bawah ini adalah salah satu procedure pada VGAFONT.TPU. Procedure ini berguna untuk menampilkan tulisan pada layar monitor. Adapun sintaksnya adalah sebagai berikut:

Sintaks	:	PutString (PicBuff^,X,Y,Col1,Col2,Text);
Fungsi	:	Untuk menampilkan tulisan pada layar monitor.
Parameter	:	PicBuff = Virtual page yang dituju.
	X,Y	= Koordinat X dan Y dari teks.
	Col1	= Warna bayangan dari teks.
	Col2	= Warna text.
	Text	= Tulisan yang hendak Anda tampilkan.

```
    PutString(Screen^,60,13,0,15,'FIRE FIGHTER MISSION EDITOR');
    Block(Screen^,9,175,319,195,Blue);
```

Procedure WINDOWS ini berguna untuk menggambar persegi panjang dengan ukuran yang dimasukkan pada parameternya. Prosesnya tidaklah terlalu sulit, cobalah pelajari procedure WINDOWS.

```
    Windows(Screen^,4,170,315,190);
```

Procedure HOLLOW ini berguna untuk menggambar lubang pada window yang ada di layar monitor. Lubang ini digunakan sebagai tempat meletakkan tulisan karena warnanya hitam sehingga tulisan tersebut mudah dibaca. Prosesnya pun tidak sulit, coba pelajari procedure ini.

```
    Hollow(Screen^,10,175,309,185);
     PutString(Screen^,40,177,0,14,'PROGRAM & DESIGN BY AGUSTINUS
NALWAN 1994');
    Windows(Screen^,70,36,260,160);
    Hollow(Screen^,90,50,240,65);
    Hollow(Screen^,90,70,240,85);
    Hollow(Screen^,90,90,240,105);
    Hollow(Screen^,90,110,240,125);
    Hollow(Screen^,90,130,240,145);
    PutString(Screen^,120,55,LightRed,14,'1. EDIT SHIP');
    PutString(Screen^,120,75,LightRed,14,'2. EDIT WEAPON');
    PutString(Screen^,120,95,LightRed,14,'3. EDIT SCENARIO');
    PutString(Screen^,120,115,LightRed,14,'4. SAVE MISSION');
    PutString(Screen^,120,135,LightRed,14,'5. QUIT');
  End;
```

Tahap menu utama telah kita selesaikan. Kelanjutan dari program setelah kita memilih salah satu option. Lihatlah procedure EXECUTE di bawah ini. Sesuai dengan Option yang dipilih, program akan memanggil procedure yang berkenaan dengan option tersebut.

```
Procedure Execute;
  Begin
    Case Option of
      1:Edit_Ship;
      2:Edit_Weapon;
      3:Edit_Scenario;
      4:If Yes_No('SAVE') then Save_Mission;
      5:If Yes_No('EXIT') then Luar:=True;
    End;
  End;
```

Proses Edit Ship

Proses pada procedure ini cukup panjang. Proses ini berguna untuk memasukkan data dari pesawat ke dalam variabel Bank_Ship. Data tersebut seperti jumlah power yang dimiliki oleh sebuah pesawat, jumlah frame yang dimiliki oleh pesawat, posisi pesawat pada tiap-tiap framenya dan lain sebagainya. Karena kita perlu memasukkan data, maka mau tidak mau kita harus membuat menu lagi dimana kita dapat menginputkan data melalui keyboard dengan mengetik. Di sini tentu saja kita tidak dapat memakai READLN karena kita sekarang bekerja di mode graphic. Untuk itu kita harus membuat READLN sendiri yang bermodus graphic. Ingat tidak semua data dimasukkan melalui keyboard dengan mengetikkan suatu nilai secara manual. Kita harus mengisi posisi pesawat (Koordinat X dan Y) pada tiap framenya. Apabila proses ini kita lakukan dengan mengetik, tidak effisien. Selain itu prosesnya pasti sangat sulit karena kita harus membayangkan koordinat pesawat tersebut pada saat kita mengetikkan koordinatnya. Di sini kita harus membuat fasilitas di mana kita dapat menggerakkan pesawat tersebut secara langsung pada tempat yang kita inginkan. Setelah kita meletakkan pesawat tersebut, editor secara otomatis akan memasukkan koordinat X dan Y ke dalam variabel Bank_Ship. Tentu saja hal ini akan lebih mempermudah proses penyuntingan.

```
Procedure Edit_Ship;
Var Luar:Boolean;
    Xpo,Ypo:Integer;
    Ix,Iy:Integer;
    Xps,Yps:Integer;
  Begin
```

Procedure SHIP_BACKGROUND dipanggil untuk melakukan penggambaran menu dari Edit Ship.

```
    Ship_Background;
```

Procedure EDIT_SHIP_STATUS dipanggil untuk melakukan penulisan status seperti Nomor pesawat yang sedang disunting, frame yang sedang disunting, gambar dari pesawat yang sedang disunting dan sebagainya.

```
    Edit_Ship_Status(1);
    Luar:=False;
    Repeat
      Repeat
```

```
Until Keypressed;
Ch:=Readkey;
Beep;
With Bank_Ship[Editor.Shp_No]^.Motion[Editor.Mot_Ctr] do
  Begin
    Xpo:=Xp;
    Ypo:=Yp;
    Case Ch of
      #9:Begin
```

Jika tombol TAB ditekan, panggil procedure EDIT_SHIP_MENU. Fungsi dari procedure ini adalah untuk menampilkan menu pengisian data dan memberikan kesempatan pada user untuk mengisikan data dari pesawat. Agar lebih jelasnya, sebaiknya Anda mencoba terlebih dahulu FIRE FIGHTER MISSION EDITOR ini.

```
        Ship_Background;
        Edit_Ship_Menu;
        Ship_Background;
      End;
```

Tombol-tombol di bawah ini yaitu mulai #75 hingga '8' berguna untuk melakukan pemindahan posisi pesawat pada frame ini. Pemindahan ini hanya dapat dilakukan bila mode pemindahan pesawat sedang dalam kondisi aktif (Bila pesawat yang sedang Anda sunting terihat dikelilingi persegi panjang putih).

```
#75:If Editor.Tag then Dec(Xp);
#77:If Editor.Tag then Inc(Xp);
#80:If Editor.Tag then Inc(Yp);
#72:If Editor.Tag then Dec(Yp);
'4':If Editor.Tag then Dec(Xp,10);
'6':If Editor.Tag then Inc(Xp,10);
'2':If Editor.Tag then Inc(Yp,10);
'8':If Editor.Tag then Dec(Yp,10);
'a','A':If Editor.Tag then
          If Yes_No('PREVIOUS LOOP ANIM') then
           Begin
             Y:=1;
```

Proses ini dilakukan pada saat user menekan 'A' yang berarti proses AUTO USER ANIM dijalankan. Prosesnya dilakukan dengan menyalin isi PIC_NO dari frame yang diulang ke frame-frame mulai frame di mana Anda berada. Perhatikan baris dimana terdapat perintah If Y=Editor.Mot_Ctr then Y:=1. Ini berfungsi untuk mengembalikan nilai Y

ke 1. PIC_NO yang diisikan dikembalikan lagi ke nilai PIC_NO pada frame nomor 1. Editor.Mot_Ctr berisi frame di mana Anda berada.

```
For X:=Editor.Mot_Ctr to Bank_Ship[Editor.Shp_No]^.Tot_Mot do
                          Begin

Bank_Ship[Editor.Shp_No]^.Motion[X].Pic_No:=

Bank_Ship[Editor.Shp_No]^.Motion[Y].Pic_No;
                            Inc(Y);
                            If Y=Editor.Mot_Ctr then Y:=1;
                          End;
                      End;
          'p','P':If Editor.Tag then
                      Begin
                        If Editor.Mot_Ctr<>1 then
                          Begin
```

Proses ini dilakukan untuk mengcopy koordinat X dan Y dari frame sebelumnya ke frame dimana Anda berada.

```
Xp:=Bank_Ship[Editor.Shp_No]^.Motion[Editor.Mot_Ctr-1].Xp;

Yp:=Bank_Ship[Editor.Shp_No]^.Motion[Editor.Mot_Ctr-1].Yp;
                          End;
                      End;
            'y','Y':If (Editor.Tag) and (Editor.Mot_Ctr>1) then
                      Begin
```

Proses ini dilakukan untuk mengambil perpindahan koordinat Y dari frame sebelumnya ke frame di mana Anda berada. Harga tersebut dimasukkan pada variabel Ly. Perpindahan koordinat ini akan dipakai untuk mengubah posisi pesawat pada frame-frame selanjutnya sehingga pada setiap frame berikutnya pesawat akan terus berpindah sejauh Ly.

```
Yps:=Bank_Ship[Editor.Shp_No]^.Motion[Editor.Mot_Ctr].Yp;
                                                        Iy:=Yps-
Bank_Ship[Editor.Shp_No]^.Motion[Editor.Mot_Ctr-1].Yp;
If Yes_No('REPEAT POSSITION') then
For X:=Editor.Mot_Ctr+1 to Bank_Ship[Editor.Shp_No]^.Tot_Mot do
With Bank_Ship[Editor.Shp_No]^.Motion[X] do
Begin
Inc(Yps,Iy);
Yp:=Yps;
```

```
End;
End;
```

Proses ini sama dengan proses di atas, bedanya sekarang kita memproses perpindahan koordinat X, bukannya koordinat Y lagi.

```
'x','X':If (Editor.Tag) and (Editor.Mot_Ctr>1) then
Begin
Xps:=Bank_Ship[Editor.Shp_No]^.Motion[Editor.Mot_Ctr].Xp;
Ix:=Xps-Bank_Ship[Editor.Shp_No]^.Motion[Editor.Mot_Ctr-1].Xp;
If Yes_No('REPEAT POSSITION') then
For X:=Editor.Mot_Ctr+1 to Bank_Ship[Editor.Shp_No]^.Tot_Mot do
With Bank_Ship[Editor.Shp_No]^.Motion[X] do
Begin
Inc(Xps,Ix);
Xp:=Xps;
End;
End;
```

Jika Space Bar ditekan mode frame diset ke INDIV atau FOLLOW.

```
' ':Editor.Poss_Mode:=Not Editor.Poss_Mode;
```

Jika Enter ditekan maka Editor.Tag yang menunjukkan mode pemindahan pesawat akan diaktifkan/dinonaktifkan.

```
#13:Editor.Tag:=Not Editor.Tag;
#27:If Yes_No('EXIT TO MAIN MENU') then Luar:=True;
'>','.':Inc(Editor.Mot_Ctr);
'<',',':Dec(Editor.Mot_Ctr);
']','}':Begin
```

Maju ke nomor pesawat berikutnya.

```
            Inc(Editor.Shp_No);
            Editor.Mot_Ctr:=1;
          End;
'[','{':Begin
```

Mundur ke nomor pesawat sebelumnya.

```
Dec(Editor.Shp_No);
Editor.Mot_Ctr:=1;
end;
End;
```

```
With Editor do
Begin
If Mot_Ctr>Bank_Ship[Shp_No]^.Tot_Mot then Mot_Ctr:=1;
If Mot_Ctr<1 then Mot_Ctr:=Bank_Ship[Shp_No]^.Tot_Mot;
If Shp_No>100 then Shp_No:=1;
If Shp_No<1 then Shp_No:=100;
End;
```

Jika Mode frame = FOLLOW maka pindahkan koordinat X dan Y dari frame berikutnya sejauh perpindahan yang dilakukan pada frame saat ini.

```
        If Editor.Poss_Mode then
For X:=Editor.Mot_Ctr+1 to Bank_Ship[Editor.Shp_No]^.Tot_Mot do
            Begin
              Inc(Bank_Ship[Editor.Shp_No]^.Motion[X].Xp,Xp-Xpo);
              Inc(Bank_Ship[editor.Shp_No]^.Motion[X].Yp,Yp-Ypo);
            End;
        End;
      Edit_Ship_Status(1);
    Until Luar;
  End;
```

Procedure EDIT_SHIP_STATUS yang berfungsi untuk menampilkan status adalah sebagai berikut.

```
Procedure Edit_Ship_Status(Tipe:Byte);
  Begin
```

Jika menu pengisian data sedang non-aktif maka tampilkan tulisan dibawah ini. Tulisan ini berfungsi untuk memberikan petunjuka pada user, tombol-tombol yang dapat ditekan pada saat itu.

```
If Tipe = 1 then
Begin
Block(Screen^,6,6,314,194,0);
PutString(Screen^,15,9,0,15,'ENTER SPACE TAB ESC X Y P A < > ] [ CURSOR');
End Else
Block(Screen^,6,6,199,180,0);
```

Tampilkan koordinat X dan Y dari pesawat.

```
PutString(Screen^,15,19,0,14,'X:
'+Strig(Bank_Ship[Editor.Shp_No]^.Motion[Editor.Mot_Ctr].Xp));
PutString(Screen^,60,19,0,14,'Y:
'+Strig(Bank_Ship[Editor.Shp_No]^.Motion[Editor.Mot_Ctr].Yp));
    With Editor do
```

```
Begin
  With Bank_Ship[Shp_No]^ do
    Begin
      If (Motion[Mot_Ctr].Pic_No<>0) then
        If Tipe = 1 then
          Begin
```

Jika menu pengisian data tidak aktif, letakkan sprite pesawat pada posisi sebenarnya.

```
PutImage(Screen^,Motion[Mot_Ctr].Xp,Motion[Mot_Ctr].Yp,Picture[
                  Motion[Mot_Ctr].Pic_No].Picture^);
```

Jika mode peletakan gambar sedang aktif tampilkan persegi panjang di sekeliling pesawat.

```
If Tag then
Rectangle(Screen^,Motion[Mot_Ctr].Xp,Motion[Mot_Ctr].Yp,
Motion[Mot_Ctr].Xp+Picture[Motion[Mot_Ctr].Pic_No].Length,
Motion[Mot_Ctr].Yp+Picture[Motion[Mot_Ctr].Pic_No].Width,15);
```

Jika ada senjata yang ditembakkan maka letakkan gambar senjata.

```
If Motion[Mot_Ctr].Fire<>0 then
Begin
If Bank_Weapon[Motion[Mot_Ctr].Fire].Pic_No<>0 then
PutImage(Screen^,Motion[Mot_Ctr].Xp+Picture[Motion[Mot_Ctr].Pic_No].Length Div 2,
Motion[Mot_Ctr].Yp+Picture[Motion[Mot_Ctr].Pic_No].Width,
Picture[Bank_Weapon[Motion[Mot_Ctr].Fire].Pic_No].Picture^);
End;
End Else
Begin
```

Jika menu pengisian data sedang aktif maka tampilkan sprite pesawat pada koordinat (50,50).

```
PutImage(Screen^,50,50,Picture[Motion[Mot_Ctr].Pic_No].Picture^);
```

Jika ada senjata yang ditembakkan, letakkan pula gambar senjata.

```
If Motion[Mot_Ctr].Fire<>0 then
Begin
If Bank_Weapon[Motion[Mot_Ctr].Fire].Pic_No<>0 then
PutImage(Screen^,50+Picture[Motion[Mot_Ctr].Pic_No].Length Div 2,
                 50+Picture[Motion[Mot_Ctr].Pic_No].Width,
```

```
Picture[Bank_Weapon[Motion[Mot_Ctr].Fire].Pic_No].Picture^);
                    End;
                End;
            End;
        Block(Screen^,10,185,310,194,0);
```

Tampilkan nomor pesawat yang sedang disunting, nomor frame yang sedang disunting, nomor senjata yang ditembakkan dan mode frame.

```
PutString(Screen^,10,185,0,15,'SHP NO: '+Strig(Shp_No)+'/100');
PutString(Screen^,110,185,0,14,'MOT NO: '+Strig(Mot_Ctr)+'/'+
Strig(Bank_Ship[Shp_No]^.Tot_Mot));
PutString(Screen^,210,185,0,14,'FIRE:
'+Strig(Bank_Ship[Shp_No]^.Motion[Mot_Ctr].Fire));
If Editor.Poss_Mode then
PutString(Screen^,270,185,0,14,'FOLLOW') else
PutString(Screen^,270,185,0,14,'INDIV ');
End;
Rectangle(Screen^,5,5,315,195,210);
Rectangle(Screen^,6,6,314,194,214);
Rectangle(Screen^,0,0,319,199,214);
Rectangle(Screen^,1,1,318,198,210);
Rectangle(Screen^,2,2,317,197,205);
Rectangle(Screen^,3,3,316,196,205);
Rectangle(Screen^,4,4,315,195,205);
End;
```

Procedure EDIT_SHIP_MENU ini dipanggil pada saat Anda menekan tombol TAB. Pertama-tama Anda akan mendapati tampilan menu pengisian data.

```
Procedure Edit_Ship_Menu;
Var Luar  : Boolean;
    Option: Byte;
  Begin
    Option:=1;
    Luar:=False;
    Menu_Background;
```

Procedure DISPLAY_EDIT_MENU_DATA ini berfungsi untuk menampilkan data dari pesawat pada menu pengisian data. Data itu meliputi jumlah power dari pesawat, nomor pesawat yang Anda sunting, jumlah frame yang dimiliki, nomor senjata yang ditembakkan pada frame ini dan lain sebagainya.

```
    Display_Edit_Menu_Data;
```

Procedure EDIT_SHIP_STATUS dipanggil dengan parameter 2 yang berfungsi untuk menampilkan status, seperti nomor sprite pesawat yang dipergunakan dan nomor senjata yang ditembakkan.

```
    Edit_Ship_Status(2);
    Repeat
      Repeat
Rectangle(Screen^,207,17+(Option-1)*15,308,14+(Option)*15-6,7);
        Delay(5);
Rectangle(Screen^,207,17+(Option-1)*15,308,14+(Option)*15-6,0);
      Until Keypressed;
      Ch:=Readkey;
      Beep;
      Case Ch of
```

Tombol kursor atas dan bawah untuk memilih field mana yang hendak diganti. Tombol kursor kiri dan kanan berfungsi untuk pindah ke nomor pesawat berikutnya atau sebelumnya.

```
        #72  :Dec(Option);
        #75  :Begin
                 Dec(Editor.Shp_No);
                 If Editor.Shp_No>200 then Editor.Shp_No:=1;
                 If Editor.Shp_No<1 then Editor.Shp_No:=200;
                 Menu_Background;
                 Display_Edit_Menu_Data;
               End;
        #77  :Begin
                 Inc(Editor.Shp_No);
                 If Editor.Shp_No>200 then Editor.Shp_No:=1;
                 If Editor.Shp_No<1 then Editor.Shp_No:=200;
                 Menu_Background;
                 Display_Edit_Menu_Data;
               End;
        #80  :Inc(Option);
```

Jika tombol Enter ditekan, lakukan proses pengisian data pada field yang dikehendaki.

```
        #13  :Fill_Data(Option);
        #27,#9:Luar:=True;
      End;
      If Option<1 then Option:=8;
      If Option>8 then Option:=1;
      Edit_Ship_Status(2);
    Until Luar;
  End;
```

Procedure FILL_DATA ini dipanggil bila Anda menekan Enter pada procedure EDIT_SHIP_MENU. Fungsinya adalah untuk mengisi data Bank_Ship pada nomor pesawat yang sedang disunting. Variabel Option sebagai parameter berfungsi untuk menentukan field mana yang akan diubah, apakah SHP_NO (Nomor pesawat yang disunting), apakah TOT_MOT (Jumlah frame) atau yang lain-lainnya.

Untuk melakukan proses pemasukan data, dipanggil function SINPUT yang dirancang sedemikian rupa sehingga dapat menggantikan fungsi dari READLN. Adapun sintaksnya adalah sebagai berikut .

Sintaks :

New_Value:=SInput(X,Y,Total_Digit,Code,Min,Max,Old_Value);

Fungsi : Sebagai READLN bermodus graphic.

Parameter :

X,Y	=	Koordinat dimana data input akan ditampilkan.
Total_Digit	=	Jumlah character yang bisa dimasukkan.
Code	=	Tipe data yang bisa dimasukkan.
Min	=	Harga minimum yang bisa dimasukkan.
Max	=	Harga maximum yang bisa dimasukkan.
Old_Value	=	Harga lama sebelum diubah.

```
Procedure Fill_Data(Option:Byte);
  Begin
    With Editor do
      With Bank_Ship[Shp_No]^ do
        Case Option of
         1:Begin
              Shp_No:=SInput(270,17,3,0,1,100,Shp_No);
              Editor.Mot_Ctr:=1;
           End;
         2:Tot_Mot:=SInput(270,32,3,0,1,200,Tot_Mot);
         3:Tot_Rep:=SInput(270,47,2,0,1,20,Tot_Rep);
         4:Energy:=SInput(270,62,3,0,0,255,Energy);
         5:Motion[Mot_Ctr].Fire:=SInput(270,77,2,0,0,20,Motion[Mot_Ctr].Fire);
      6:Motion[Mot_Ctr].Pic_No:=SInput(270,92,3,0,1,255,Motion[Mot_Ctr].Pic_No);
```

Lakukan proses AUTO ANIM.

```
          7:Auto_Arrange;
```

Panggil procedure PREVIEW_SHIP untuk menggerakkan pesawat sesuai routenya.

```
          8:Preview_Ship(Shp_No);
        End;
    Menu_Background;
    Display_Edit_Menu_Data;
    Edit_Ship_Status(2);
  End;

Procedure Preview_Ship(Shp_No:Byte);
```

Procedure ini berfungsi untuk menjalankan sprite pesawat menurut routenya yaitu dengan meletakkan sprite pesawat secara berurutan pada koordinat yang disimpan dalam setiap framenya.

```
  Begin
    Block(Screen^,0,0,319,199,0);
      With Bank_Ship[Shp_No]^ do
        Begin
```

Variabel X berfungsi sebagai petunjuk nomor frame.

```
          X:=1;
          While (Motion[X].Pic_No<>0) and (X<=Tot_Mot) do
            Begin
```

Letakkan sprite pesawat pada posisi yang diinstruksikan di dalam frame (Yaitu koordinat Motion[X].Xp, Motion.[X].Yp).

```
              PutImage(Screen^,Motion[X].Xp,Motion[X].Yp,
              Picture[Motion[X].Pic_No].Picture^);
              Delay(40);
```

Hapus sprite pesawat. Di sini kita tidak menggunakan teknik animasi tiga layar karena masalah tampilan pada pembuatan editor tidaklah terlalu penting, apalagi fungsinya cuma untuk mengisikan data, hal ini tidak mempengaruhi animasi game engine yang akan kita buat.

```
              Block(Screen^,Motion[X].Xp,Motion[Y].Yp,Motion[X].Xp+
              Picture[Motion[X].Pic_No].Length,
              Motion[X].Yp+Picture[Motion[X].Pic_No].Width,0);
```

Tambahkan variabel X dengan 1 yang berarti ke frame selanjutnya.

```
            Inc(X);
          End;
      End;
    Menu_Background;
  End;
```

Proses Edit Weapon

Pada bagian ini, program bertugas untuk memasukkan data dari senjata. Tentu saja bagian ini jauh lebih sederhana dibandingkan dengan bagian EDIT SHIP mengingat tidak diperlukannya fasilitas peletakan sprite pesawat dengan menggerakkannya secara langsung. Peluru tidak memiliki posisi yang tetap. Posisinya ditentukan di mana dia ditembakkan. Untuk itu kita cukup menyimpan data kecepatan sumbu X dan kecepatan sumbu Y dari setiap peluru.

```
Procedure Edit_Weapon;
Var Luar  : Boolean;
    Option: Byte;
  Begin
    Option:=1;
    Luar:=False;
```

Tampilkan menu dari proses EDIT WEAPON.

```
    Menu_Weapon_Background;
```

Tampilkan data dari senjata pada layar monitor.

```
    Display_Weapon_Menu_Data;
```

Tampilkan status dari proses EDIT WEAPON. Pada procedure EDIT_WEAPON_STATUS ini dilakukan peletakan sprite senjata pada layar monitor.

```
    Edit_Weapon_Status;
    Repeat
      Repeat
Rectangle(Screen^,207,17+(Option-1)*15,308,14+(Option)*15-6,7);
        Delay(15);
Rectangle(Screen^,207,17+(Option-1)*15,308,14+(Option)*15-6,0);
      Until Keypressed;
      Ch:=Readkey;
      Beep;
```

```
      Case Ch of
        #72   :Dec(Option);
        #80   :Inc(Option);
```

Panggil procedure FILL_WEAPON_DATA untuk melakukan pengisian data apabila tombol Enter ditekan.

```
        #13   :Fill_Weapon_Data(Option);
        #27   :If Yes_No('EXIT TO MAIN MENU') then Luar:=True;
      End;
      If Option<1 then Option:=8;
      If Option>8 then Option:=1;
      Edit_Weapon_Status;
    Until Luar;
  End;

Procedure Edit_Weapon_Status;
```

Procedure ini dipanggil untuk menampilkan sprite dari senjata yang sedang disunting.

```
  Begin
    With Editor do
      With Bank_Weapon[Wpn_No] do
        Begin
          Block(Screen^,50,100,150,198,0);
                                              If    Pic_No<>0    then
PutImage(Screen^,50,100,Picture[Pic_No].Picture^);
        End;
  End;

Procedure Fill_Weapon_Data(Option:Byte);
```

Procedure ini dipanggil pada saat Anda menekan tombol Enter. Di sini dilakukan pengisian data pada field dalam variabel Bank_Weapon.. Sama seperti proses pengisian data pada EDIT SHIP, dipergunakan pula function SINPUT.

```
  Begin
    With Editor do
      With Bank_Weapon[Wpn_No] do
        Case Option of
          1:Wpn_No:=SInput(270,17,2,0,1,20,Wpn_No);
          2:Vx:=SInput(270,32,4,0,-200,200,Vx);
          3:Vy:=SInput(270,47,4,0,-200,200,Vy);
          4:Destruct:=SInput(270,62,3,0,0,255,Destruct);
```

```
5:Tipe:=SInput(270,77,1,0,0,10,Tipe);
6:Tot_Rep:=SInput(270,92,2,0,1,20,Tot_Rep);
7:Pic_No:=SInput(270,107,3,0,1,255,Pic_No);
```

Bila option yang dipilih nomor 8 maka panggil procedure PREVIEW_WEAPON. Pada procedure ini sprite senjata akan dijalankan sesuai dengan routenya (Karena Bank_Weapon tidak memiliki frame maka perubahan posisi dilakukan berdasarkan kecepatan, bukan berdasarkan posisi tiap framenya).

```
        8:Preview_Weapon;
      End;
  Menu_Weapon_Background;
```

Tampilkan kembali data yang telah diperbarui ke layar monitor.

```
  Display_Weapon_Menu_Data;
End;
```

Procedure ini dipanggil untuk menjalankan senjata yang sedang disunting sesuai routenya.

```
Procedure Preview_Weapon;
Var Xp,Yp:Integer;
  Begin
    Block(Screen^,1,1,318,198,0);
    Xp:=155; Yp:=0;
```

Variabel Editor.Frm_Ctr berfungsi sebagai counter sprite yang dipakai oleh senjata tersebut.

```
    Editor.Frm_Ctr:=0;
    With Editor do
      With Bank_Weapon[Wpn_No] do
        Repeat
          Inc(Frm_Ctr);
```

Kalau counter sprite sudah sama atau melebihi jumlah sprite yang harus diulang maka kembalikan nilai Frame_Ctr ke 0. Dengan cara ini maka akan terjadi perulangan sprite yang akan ditampilkan pada layar monitor.

```
        If Frm_Ctr>=Tot_Rep then Frm_Ctr:=0;
```

Letakkan sprite senjata ke layar monitor.

```
PutImage(Screen^,Xp,Yp,Picture[Pic_No+Frm_Ctr].Picture^);
          Delay(50);
```

Hapus sprite senjata pada layar monitor. Teknik yang kita gunakan di sini masih teknik gambar-hapus. Memang teknik ini kurang begitu bagus dibandingkan teknik animasi tiga layar. Cara ini digunakan mengingat kita masih dalam tahap membuat editor. Tapi jangan sekali-kali memakai teknik ini pada game engine.

```
          Block(Screen^,Xp,Yp,Xp+Picture[Pic_No+Frm_Ctr].Length,
          Yp+Picture[Pic_No+Frm_Ctr].Width,0);
```

Tambah koordinat X dan Y dengan kecepatan sumbu X dan kecepatan sumbu Y.

```
          Inc(Xp,Vx);
          Inc(Yp,Vy);
          Rectangle(Screen^,0,0,319,199,LightBlue);
```

Loop sampai koordinat dari senjata telah melampaui batas layar.

```
Until (Xp<0) or (Xp+Picture[Pic_No+Frm_Ctr].Length>319) or (Yp<0)
or
(Yp+Picture[Pic_No+Frm_Ctr].Width>199) or (Tipe=2);
  End;
```

Proses Edit Scenario

Pada proses edit skenario ini dilakukan penyusunan level/skenario dari permainan. Di sini editor harus menyediakan fasilitas untuk mengisikan nomor pesawat yang hendak dikeluarkan pada waktu yang dapat ditentukan pula. Untuk itu kita masih tetap menggunakan menu untuk pengisian data. Dan yang paling penting adalah pemberian fasilitas preview dari pesawat yang hendak dimunculkan pada saat itu dan juga preview dari skenario. Tujuan pemberian fasilitas preview dari pesawat adalah untuk mengetahui route dari pesawat tersebut tanpa harus masuk ke EDIT SHIP terlebih dahulu. Sedangkan tujuan diberikannya fasilitas preview dari skenario adalah untuk menjalankan skenario yang telah Anda rancang sehingga Anda dapat melakukan evaluasi tanpa perlu menjalankan skenario tersebut melalui game engine.

```
Procedure Edit_Scenario;
Var Luar  : Boolean;
    Option: Byte;
  Begin
    Option:=1;
    Luar:=False;
```

Tampilkan background, menu, status dan juga data dari proses edit skenario. Adapun yang dimaksud dengan data dari edit skenario ialah nomor skenario yang sedang Anda sunting, nomor pesawat yang hendak dimunculkan dan waktu di mana pesawat tersebut akan dimunculkan.

```
    Scenario_Background;
    Scenario_Status;
    Display_Scenario_Menu_Data;
    Repeat
      Repeat
Rectangle(Screen^,207,17+(Option-1)*15,308,14+(Option)*15-6,7);
        Delay(15);
Rectangle(Screen^,207,17+(Option-1)*15,308,14+(Option)*15-6,0);
      Until Keypressed;
      Ch:=Readkey;
      Beep;
      Case Ch of
        #72  :Dec(Option);
        #80  :Inc(Option);
```

Jika tombol Enter ditekan lakukan proses pengisian data.

```
        #13  :Fill_Scenario_Data(Option);
        #27  :If Yes_No('EXIT TO MAIN MENU') then Luar:=True;
      End;
      If Option<1 then Option:=5;
      If Option>5 then Option:=1;
      Scenario_Background;
      Scenario_Status;
      Display_Scenario_Menu_Data;
    Until Luar;
  End;
```

Pada procedure ini dilakukan pengisian data pada field dari Scenario

```
Procedure Fill_Scenario_Data(Option:Byte);
Begin
    With Editor do
      Case Option of
        1:Scene_No:=SInput(270,17,3,0,1,5000,Scene_No);
```

```
2:Scenario[Scene_No].Shp_No:=SInput(270,32,3,0,0,100,Scenario[Scene_No].Shp_No);
3:Scenario[Scene_No].Timer:=SInput(270,47,5,0,1,30000,Scenario[Scene_No].Timer);
```

Lakukan pemanggilan procedure PREVIEW_SHIP untuk menjalankan sprite pesawat sesuai routenya.

```
4:Preview_Ship(Scenario[Scene_No].Shp_No);
```

Jalankan skenario yang telah disunting.

```
        5:Preview_Scenario;
      End;
    Scenario_Background;
    Display_Scenario_Menu_Data;
  End;
```

Proses Preview Scenario

Proses ini dilakukan pada saat Anda memilih PREVIEW SCENE pada menu dari EDIT SCENARIO. Hampir secara keseluruhan, proses yang dilakukan di sini sama persis dengan game engine yang hendak kita buat. Karena tujuan dari preview ini adalah untuk melihat hasil jadi dari skenario/level yang kita rancang. Cuma tidak ditampilkan pesawat dari pemain.

```
Procedure Preview_Scenario;
Var Luar,No_Data:Boolean;
  Begin
    Fill_Page(Proc^,0);
```

Isi semua variabel yang diperlukan dengan nilai awal.

```
    Refresh_Data;
    Luar:=False;
```

Variabel No_Data akan berharga TRUE kalau sudah tidak ada lagi pesawat yang perlu ditampilkan pada skenario.

```
    No_Data:=False;
```

Baris scroll page kita isi dengan 599. Variabel ini nantinya akan kita kurangi 1 pada setiap loop sehingga layar background akan bergeser.

Teknik ini sudah tidak asing lagi bagi kalian (apabila Anda telah membaca penjelasan mengenai scroll page pada buku ini).

```
Scroll:=599;
Repeat
  Repeat
```

Variabel Loop ditambah dengan satu. Apabila Loop telah = Loop_Time maka proses dibawah baru dilaksanakan. Ini dilakukan agar kecepatan dari permainan selalu sama walaupun dijalankan pada microprocessor yang berkecepatan berbeda pula. Loop_Time seperti yang telah dijelaskan di atas pada bagian pendeteksian kecepatan CPU, adalah variabel yang menampung keterangan mengenai berapa kali looping harus dilakukan sebelum animasi berganti ke frame selanjutnya. Disarankan bagi Anda untuk tidak memakai DELAY, fasilitas dari TURBO PASCAL, karena procedure ini akan menghasilkan waktu tunggu yang berlainan untuk microprocessor yang berbeda. DELAY(1000) pada komputer 286 jauh lebih lama dibandingkan DELAY(1000) pada komputer 486.

```
Inc(Loop);
If Loop=Loop_Time then
  Begin
```

Letakkan background page pada virtual page.

```
Put_Scroll_Page(Proc^,Scroll);
```

Kalkulasi situasi dari permainan, seperti pemindahan koordinat masing-masing pesawat, pemindahan koordinat senjata yang aktif dan lain-lain.

```
Do_Status(No_Data);
```

Lakukan pendeteksian situasi dan kondisi dari tiap object. Apakah sebuah pesawat sudah harus dinonaktifkan karena telah menjalani semua frame?

```
Do_Checking(Luar,No_Data);
```

Letakkan semua sprite yang harus ditampilkan pada layar monitor.

```
Put_Sprite;
```

Letakkan informasi status yaitu waktu, nomor skenario yang sedang dijalankan. Tentu saja hal ini tidak perlu dilakukan pada game engine.

```
            Put_Status;
```

Pindahkan virtual page ke visual page.

```
            MoveTo(Proc^,Screen^);
            Loop:=1;
          End;
      Until (Keypressed) or (Luar);
      If Not Luar then
        Begin
          Ch:=Readkey;
          Beep;
          Case Ch of
            ' ':Begin
```

Jika tombol Space Bar ditekan, lakukan PAUSE.

```
                  Ch:=Readkey;
                End;
```

Jika tombol ESC ditekan, akhiri preview scenario dan kembali ke penyuntingan menu dari skenario.

```
            #27:Luar:=True;
          End;
        End;
    Until Luar;
  End;
```

Pada procedure ini dilakukan pengisian nilai awal pada variabel-variabel yang akan dipakai.

```
Procedure Refresh_Data;
  Begin
```

Time diset ke 0. Nomor skenario diset ke 1.

```
    Editor.Time:=0;
    Editor.Scene_No:=1;
```

Isi variabel SHIP dengan 0. Isi variabel WEAPON dengan 0. Kegunaan kedua variabel ini akan dijelaskan pada bagian PROSES OBJECT STATUS.

```
    For X:=1 to 20 do
      FillChar(Ship[X],SizeOf(Rec_Ship_Status),0);
    For X:=1 to 100 do
      FillChar(Weapon[X],SizeOf(Rec_Weapon_Status),0);
  End;
```

Proses Object Status

Anda tentu saja masih ingat struktur main program dari program game yang pernah kita buat sebelumnya. Hampir semua game memiliki struktur main program yang sama. Di mana sebelum dilakukan penggambaran sprite, dilakukan dahulu pemindahan posisi dari tiap object, kemudian dilakukan pendeteksian tembakan, tabrakan, jumlah power dan lain sebagainya. Struktur inilah yang kita pakai dalam membuat game ini. Tentu saja pada program editor bagian preview skenario ini, kita harus mengikuti struktur program dari game yang akan kita buat.

```
Procedure Do_Status(Var No_Data:Boolean);
  Begin
```

Jalankan apa yang diinstruksikan pada skenario berikutnya.

```
    Do_Scenario(No_Data);
```

Pindahkan posisi setiap pesawat ke koordinat yang baru, ubah counter gambar sprite dari pesawat dan lain sebagainya.

```
    Do_Ship_Status;
```

Pindahkan posisi setiap senjata yang ditembakkan ke koordinat yang baru, ubah counter gambar sprite dari senjata. Proses ini hampir sama dengan proses di atas.

```
    Do_Weapon_Status;
```

Tambah variabel Time dengan 1. Variabel ini berfungsi sebagai waktu dari permainan. Dengan waktu ini kita dapat mengetahui kapan sebuah pesawat harus dimunculkan pada layar monitor.

```
    Inc(Editor.Time);
```

Variabel Scroll dikurangi, variabel ini berfungsi sebagai petunjuk nomor start baris dari gambar background yang harus ditampilkan dilayar monitor.

```
Dec(Scroll);
    If Scroll<1 then Scroll:=799;
  End;
```

Procedure ini adalah procedure yang memegang peranan paling penting pada proses Preview_Scene. Tugas dari procedure ini adalah mengaktifkan sebuah pesawat musuh apabila waktu (variabel Time) sama dengan waktu yang diinstruksikan dalam variabel Skenario.

```
Procedure Do_Scenario(Var No_Data:Boolean);
Begin
    While (Scenario[Editor.Scene_No].Timer=Editor.Time)
    and (Scenario[Editor.Scene_No].Timer<>0) do
      Begin
        With Scenario[Editor.Scene_No] do
          If Shp_No<>0 then
            Begin
```

Proses dibawah ini dilakukan untuk mengaktifkan sebuah pesawat baru yaitu pada variabel record SHIP. Variabel ini adalah variabel array berjumlah 20 cell. Tiap cellnya dapat menampung data dari 1 pesawat. Untuk mengaktifkan sebuah pesawat baru, dilakukan pencarian cell yang kosong (belum diisi dengan pesawat). Bila ditemukan (Active = FALSE) maka isi nomor pesawat yang mengisi cell ini dengan nomor pesawat yang diperintahkan untuk muncul pada variabel Scenario. Nomor pesawat ini menunjukkan nomor indeks dari pesawat pada variabel Bank_Ship (di mana data semua pesawat ditampung).

Kita menggunakan variabel SHIP sebagai penampung pesawat yang hendak diaktifkan, tidak langsung menggunakan variabel Bank_Ship. variabel Bank_Ship hanya menampung data dari pesawat. Data ini tidak akan diubah. Sedangkan data dari pesawat musuh yang muncul di layar dapat berubah, misalnya bila kena tembakan, powernya akan berkurang. Apabila Anda tetap ingin mengubah data pada Bank_Ship maka satu jenis pesawat hanya dapat ditampilkan satu kali. Karena Anda telah kehilangan data besarnya power awal (apabila pesawat ini telah kena tembak, power akan berkurang). Dan lagi selama

pesawat ini masih muncul di layar, Anda tidak dapat memunculkan pesawat jenis ini lagi karena untuk pesawat jenis ini, Anda hanya mempunyai tempat penampungan data satu cell saja pada Bank_Ship. Untuk itulah maka digunakan variabel baru yang berfungsi seperti tempat pemesanan. Setiap ada pesawat baru hendak dimunculkan, kita cukup memesan tempat pada variabel ini dan memberi informasi pesawat jenis apa yang memesan tempat tersebut. Segala perubahan nilai seperti perubahan jumlah power dilakukan pada variabel Ship ini, jadi variabel Bank_Ship tidak akan pernah diubah.

```
For X:=1 to 20 do
  With Ship[X] do
    If Not Active then
      Begin
```

Aktifkan tempat penyimpanan pesawat/variabel SHIP.

```
Active:=True;
```

Beri informasi nomor pesawat yang memesan tempat.

```
Ship[X].Shp_No:=Scenario[Editor.Scene_No].Shp_No;
Ctr:=1;
```

Isikan jumlah power awal dari pesawat. Data ini tentu saja didapat dari variabel Bank_Ship dengan indeks nomor pesawat yang memesan variabel SHIP[X] ini.

```
Energy:=Bank_Ship[Ship[X].Shp_No]^.Energy;
```

Isi X dengan 20 untuk keluar dari looping.

```
X:=20;
End;
```

Tanbahkan nomor skenario.

```
Inc(Editor.Scene_No);
End;
End;
```

Jika sudah tidak ada pesawat yang harus ditampilkan maka isi No_Data dengan TRUE.

```
    If Scenario[Editor.Scene_No].Shp_No=0 then No_Data:=True;
  End;

Procedure Do_Ship_Status;
```

Procedure ini berfungsi untuk melakukan perubahan status dari masing-masing pesawat yang sedang aktif. Perubahan itu meliputi perubahan koordinat, perubahan nomor sprite yang perlu diulang dan lain sebagainya.

```
Begin
  For X:=1 to 20 do
    With Ship[X] do
      If Active then
        Begin
          Inc(Ctr);
```

Jika ada senjata yang ditembakkan maka lakukan proses penembakan senjata.

```
          If Bank_Ship[Shp_No]^.Motion[Ctr].Fire<>0 then
            Begin
```

Sistem dari pengaktifan sebuah senjata pada layar permainan ini sama dengan pengaktifan sebuah pesawat musuh pada layar permainan. Digunakan variabel Weapon sebagai tempat penampungan dari senjata yang ditembakkan.

```
              For Y:=1 to 100 do
                With Weapon[Y] do
```

Wpn_No=0 berarti belum ada senjata yang memesan tempat ini (Weapon[Y]).

```
                  If Wpn_No=0 then
                    Begin
```

Isi informasi nomor senjata yang memesan tempat ini.

```
Weapon[Y].Wpn_No:=Bank_Ship[Shp_No]^.Motion[Ship[X].Ctr].Fire;
```

Isi koordinat awal di mana senjata akan muncul. Perhitungan di bawah ini adalah perhitungan titik tengah dari pesawat musuh. Sehingga senjata yang akan dimunculkan nantinya akan berada pada posisi tengah-tengah dari pesawat yang menembakkannya.

```
Weapon[Y].Xp:=Bank_Ship[Shp_No]^.Motion[Ship[X].Ctr].Xp+(Picture[
Bank_Ship[Shp_No]^.
Motion[Ship[X].Ctr].Pic_No].Length Div 2);
Weapon[Y].Yp:=Bank_Ship[Shp_No]^.Motion[Ship[X].Ctr].Yp+Picture[B
ank_Ship[Shp_No]^.
Motion[Ship[X].Ctr].Pic_No].Width;
Weapon[Y].Ctr:=1;
```

Keluar dari looping.

```
                    Y:=100;
                  End;
            End;
          End;
  End;
```

Procedure ini berguna untuk memindahkan posisi dari senjata yang sedang aktif pada layar monitor.

```
Procedure Do_Weapon_Status;
Begin
    For X:=1 to 100 do
      With Weapon[X] do
        If Wpn_No<>0 then
          Begin
            Inc(Ctr);
```

Pindahkan koordinat dari senjata berdasarkan kecepatannya.

```
            Inc(Xp,Bank_Weapon[Wpn_No].Vx);
            Inc(Yp,Bank_Weapon[wpn_No].Vy);
```

Jika koordinat dari senjata sudah keluar dari layar monitor maka senjata akan dinonaktifkan.

```
If (Xp+Picture[Bank_Weapon[Wpn_No].Pic_No].Length>319) or (Xp<0)
or (Yp+Picture[Bank_Weapon[Wpn_No].Pic_No].Width>199) or (Yp<0)
then Wpn_No:=0;
End;
End;
```

Proses Checking

Proses berikutnya adalah proses pendeteksian situasi dari object. Apabila sebuah pesawat musuh sudah menjalani semua framenya maka pesawat tersebut harus dinonaktifkan.

```
Procedure Do_Checking(Var Luar:Boolean;No_Data:Boolean);
Var Still_Work:Boolean;
  Begin
    Still_Work:=False;
    For X:=1 to 20 do
      With Ship[X] do
        If Active then
          Begin
            If Ctr>Bank_Ship[Shp_No]^.Tot_Mot then
              Begin
                Ctr:=1;
                Active:=False;
              End;
```

Still_Work diisi dengan TRUE. Variabel ini berfungsi untuk menunjukkan masih ada pesawat yang aktif jika harganya TRUE.

```
              Still_Work:=True;
            End;
```

Preview Scenario akan diakhiri bila sudah tidak ada lagi pesawat yang hendak dikeluarkan dan sudah tidak ada pesawat yang aktif pada layar permainan.

```
    If No_Data and (Still_Work = False) then Luar:=True;
  End;
```

Proses Put Sprite

Proses ini tentu saja untuk meletakkan sprite pada virtual page. Sprite-sprite ini adalah sprite pesawat dan sprite senjata.

```
Procedure Put_Sprite;
  Begin
    For X:=1 to 20 do
```

Jika pesawat tersebut aktif, letakkan gambar pesawat. Informasi nomor sprite yang harus diletakkan berada pada variabel Bank_Ship dengan indeks Ship[X].Shp_No.

```
      If Ship[X].Active then
        With Bank_Ship[Ship[X].Shp_No]^.Motion[Ship[X].Ctr] do
          Begin
            If Pic_No<>0 then PutImage(Proc^,Xp,Yp,Picture
            [Pic_No].Picture^);
          End;
```

Jika senjata tersebut aktif, letakkan gambar senjata. Informasi nomor sprite yang harus diletakkan berada pada variabel Bank_Weapon dengan indeks Weapon[X].Wpn_No.

```
   For X:=1 to 100 do
      With Weapon[X] do
        If Wpn_No<>0 then
          Begin
            If Bank_Weapon[Wpn_No].Pic_No<>0 then
PutImage(Proc^,Xp,Yp,Picture[Bank_Weapon[Wpn_No].Pic_No].Picture^);
          End;
  End;
```

Proses Put Status

Proses ini sebenarnya cuma tambahan saja yaitu untuk menampilkan nomor skenario yang sekarang dan juga untuk menampilkan waktu dari permainan.

```
Procedure Put_Status;
  Begin
PutString(Proc^,5,180,0,14,'SCENE
'+Strig(Editor.Scene_No)+'/'+'5000');
    PutString(Proc^,120,180,0,14,'TIMER: '+Strig(Editor.Time));
  End;
```

Save Mission

Proses ini dilakukan untuk menyimpan semua data yang telah Anda edit ke dalam tiga buah file. File pertama yaitu SHIP.DAT yang berisi data dari pesawat. File kedua adalah WEAPON.DAT yang berisi data dari senjata. File yang terakhir adalah SCENARIO.DAT yang berisi nomor-nomor pesawat yang harus dikeluarkan pada waktu-waktu tertentu atau dengan kata lain menyimpan skenario dari permainan.

```
Procedure Save_Mission;
  Begin
    Assign(File_Data,'Ship.Dat');
    Rewrite(File_Data,1);
    BlockWrite(File_Data,Bank_Ship[1]^,50*Sizeof(Rec_Ship));
```

```
    BlockWrite(File_Data,Bank_Ship[51]^,50*Sizeof(Rec_Ship));
    Close(File_Data);

    Assign(File_Data,'Weapon.Dat');
    Rewrite(File_Data,1);
    BlockWrite(File_Data,Bank_Weapon[1],Sizeof(Bank_Weapon));
    Close(File_Data);

    Assign(File_Data,'Scenario.Dat');
    Rewrite(File_Data,1);
    BlockWrite(File_Data,Scenario,SizeOf(Scenario));
    Close(File_Data);
  End;
```

Anda sudah memahami program editor di atas. Kalau masih juga belum mengerti, sebaiknya Anda coba baca penjelasan di atas berulang-ulang, sampai mengerti. Bila Anda telah mengerti cara kerja program di atas terutama bagian preview skenarionya, untuk mempelajari game enginenya sangatlah mudah. Hampir 75 % sama dengan bagian preview skenario.

FIRE FIGHTER GAME ENGINE

PENDAHULUAN

Sekarang kita sudah sampai pada bagian yang paling terakhir. Di sini kita akan membahas mengenai beda antara engine dan editornya. Listing yang akan dijelaskan berikut ini adalah FIRE FIGHTER GAME ENGINE! Bila Anda telah menguasai program di bawah ini Anda akan berpikir bahwa membuat game seperti RAPTOR itu tidak sulit (tapi bukan pada bagian sound programmingnya). Game RAPTOR itu saya akui benar-benar canggih pada sound programmingya karena sistem yang dipakai adalah sistem channel dimana kita dapat mendengar delapan suara sekaligus (menggunakan proses mixing). Kita belum membahas masalah sound programming, bagaimana cara memprogram sound dengan real sound, sound blaster dan lain sebagainya. Terus terang kita tidak akan mempelajari hal-hal itu pada buku ini mengingat buku ini masih dasar. Mungkin pada edisi-edisi mendatang, sound programming akan kita bahas.

PEMBUATAN FIRE FIGHTER GAME ENGINE

Listing Program

```
{-------------------------------------------------------------}
{ File       : Fire.Pas                                       }
{ Compiler   : Turbo Pascal 7.00                              }
{ Fungsi     :  Engine dari game Fire Fighter.                }
{-------------------------------------------------------------}

Uses Crt,VgaCad,Dos;

Const Left   = 1;
      Right  = 2;
      Down   = 3;
      Up     = 4;
      Frame_Per_Sec = 35;
```

```
Type
    Rec_Motion = Record
      Xp,Yp : Integer;
      Fire  : Byte;
      Pic_No: Byte;
    End;

    Rec_Ship = Record
      Tot_Mot: Byte;
      Tot_Rep: Byte;
      Motion: Array[1..200] of Rec_Motion;
      Energy: Byte;
    End;

    Rec_Ship_Status = Record
      Shp_No: Byte;
      Ctr   : Integer;
      Active: Boolean;
      Energy: Integer;
    End;

    Rec_Weapon = Record
      Vx,Vy   : Integer;
      Pic_No  : Byte;
      Destruct: Byte;
      Tipe    : Byte;
      Tot_Rep : Byte;
    End;

    Rec_Weapon_Status = Record
      Xp,Yp : Integer;
      Vx,Vy : Integer;
      Wpn_No: Byte;
      Ctr   : Byte;
    End;

    Rec_Anim = Record
      Frame : Array[1..20] of Integer;
      Total : Integer;
    End;

    Rec_Anim_Status = Record
      Xp,Yp : Integer;
      Vx,Vy : Integer;
      Ctr   : Integer;
      Active: Boolean;
      Anm_Nb: Byte;
    End;
```

```
    Rec_Sprite = Record
      Length, Width: Integer;
      Picture      : Pointer;
    End;

    Rec_Scene = Record
      Shp_No: Byte;
      Timer : Integer;
    End;
    Sound_S = Record
      Frame: Array[1..200] of Integer;
    End;

Var Scenario     : Array[1..5000] of Rec_Scene;
    Bank_Ship    : Array[1..100]  of ^Rec_Ship;
    Ship         : Array[1..20]   of Rec_Ship_Status;
    Bank_Weapon  : Array[1..20]   of Rec_Weapon;
    Weapon       : Array[1..100]  of Rec_Weapon_Status;
    Picture      : Array[1..255]  of Rec_Sprite;
    Bank_Anim    : Array[1..5]    of Rec_Anim;
    Anim         : Array[1..200]  of Rec_Anim_Status;
    SEffect      : Array[1..5] of Sound_S;
    SBuffer      : Sound_S;
    Proc         : Pointer;
    X,Y          : Integer;
    Pal_Ctr      : Integer;
    Counter_Loop : Integer;
    Loop_Time    : Integer;
    Luar         : Boolean;
    Scroll       : Integer;
    Scene_No     : Integer;
    Time         : Longint;
    Bomb         : Integer;
    Dynamite     : Boolean;
    DynaCtr      : Byte;
    Ending       : Boolean;
    Intro        : Boolean;
    EndCtr       : Integer;
    Life         : Integer;
    MyGun        : Integer;
    Ch           : Char;
    Sound_Ctr    : Integer;
    File_Data    : File;
    Dir          : Byte;

Procedure PlaySound(No:Byte);
  Begin
    Move(SEffect[No],SBuffer,SizeOf(SBuffer));
    Sound_Ctr:=1;
  End;
```

```
Procedure Load_Data;
  Begin
    For X:=1 to 100 do
      GetMem(Bank_Ship[X],Sizeof(Rec_Ship));
    Assign(File_Data,'Ship.Dat');
    Reset(File_Data,1);
    BlockRead(File_Data,Bank_Ship[1]^,50*Sizeof(Rec_Ship));
    BlockRead(File_Data,Bank_Ship[51]^,50*Sizeof(Rec_Ship));
    Close(File_Data);

    Assign(File_Data,'Weapon.Dat');
    Reset(File_Data,1);
    BlockRead(File_Data,Bank_Weapon[1],Sizeof(Bank_Weapon));
    Close(File_Data);

    Assign(File_Data,'Scenario.Dat');
    Reset(File_Data,1);
    BlockRead(File_Data,Scenario,SizeOf(Scenario));
    Close(File_Data);
  End;

Procedure Load_Pictures;
Var File_Size: Text;
    Szs      : Integer;
  Begin
    Assign(File_Data,'Picture.Pic');
    Reset(File_Data,1);
    Assign(File_Size,'Picture.Szs');
    Reset(File_Size);
    X:=0;
    Repeat
      Inc(X);
      Readln(File_Size,Szs);
      Readln(File_Size,Picture[X].Length);
      Readln(File_Size,Picture[X].Width);
      GetMem(Picture[X].Picture,Szs);
      BlockRead(File_Data,Picture[X].Picture^,Szs);
    Until (Eof(File_Size));
    Close(File_Size);
    Close(File_Data);
    Load_Scroll_Page('Back.Pic');
  End;

Procedure Global_Var;
  Begin
    Luar:=False;
    Pal_Ctr:=1;
    Counter_Loop:=1;
    Ending:=False;
```

```
    Intro:=True;
    EndCtr:=1;
    Sound_Ctr:=1;
    GetMem(Proc,64000);
    Scroll:=799;
    Dynamite:=False;
    DynaCtr:=1;
  End;

Procedure Initialize;
  Begin
    OpenVga256;
    Install_Palette('Picture.Pal');
    Set_All_Palette(Active_Pal,0,0,0);
    Refresh_Palette;
    Load_Data;
    Load_Pictures;
    Global_Var;
    Randomize;
  End;

Procedure Firing(Gun:Byte);
  Begin
    If (Gun=5) and (Bomb<1) then else
    For Y:=1 to 19 do
      With Weapon[Y] do
        If Wpn_No=0 then
          Begin
            If Gun=5 then Dec(Bomb);
            PlaySound(1);
            Weapon[Y].Wpn_No:=Gun;

Weapon[Y].Xp:=Bank_Ship[1]^.Motion[1].Xp+(Picture[Bank_Ship[1]^.
            Motion[Ship[1].Ctr].Pic_No].Length Div 2);
            Weapon[Y].Yp:=Bank_Ship[1]^.Motion[1].Yp;
            Weapon[Y].Ctr:=1;
            Y:=19;
          End;
  End;

Procedure Put_Sprite;
Var Counter:Integer;
  Begin
    For X:=1 to 200 do
      With Anim[X] do
        If Active then
          Begin
              Case Anm_Nb of
                1,2:If Bank_Anim[Anm_Nb].Frame[Ctr]<>0 then
```

```
PutImage(Proc^,Xp,Yp,Picture[Bank_Anim[Anm_No].Frame[Ctr]].Picture^);
              3 :PutImage(Proc^,Xp,Yp,Picture[Round(Random(13)+132)].Picture^)
                End;
            End;
      For X:=20 downto 1 do
        If Ship[X].Active then
          With Bank_Ship[Ship[X].Shp_No]^.Motion[Ship[X].Ctr] do
            Begin
              If Pic_No<>0 then
                If X=1 then

PutImage(Proc^,Bank_Ship[Ship[X].Shp_No]^.Motion[1].Xp,

Bank_Ship[Ship[X].Shp_No]^.Motion[1].Yp,Picture[Pic_No].Picture^)
                Else
                  PutImage(Proc^,Xp,Yp,Picture[Pic_No].Picture^);
            End;
      For X:=1 to 100 do
        With Weapon[X] do
          If Wpn_No<>0 then
            Begin
              Counter:=Ctr Mod Bank_Weapon[Wpn_No].Tot_Rep;
              If Bank_Weapon[Wpn_No].Pic_No<>0 then
PutImage(Proc^,Xp,Yp,Picture[Bank_Weapon[Wpn_No].Pic_No+Counter].Picture^);
            End;
    End;

Procedure Do_Scenario(Var No_Data:Boolean);
    Begin
While (Scenario[Scene_No].Timer=Time) and (Scenario[Scene_No].Timer<>0) do
        Begin
          With Scenario[Scene_No] do
            If Shp_No<>0 then
              Begin
                For X:=2 to 20 do
                  With Ship[X] do
                    If Not Active then
                      Begin
                        Active:=True;
                        Ship[X].Shp_No:=Scenario[Scene_No].Shp_No;
                        Ctr:=1;
                        Energy:=Bank_Ship[Ship[X].Shp_No]^.Energy;
                        X:=20;
                      End;
                Inc(Scene_No);
              End;
        End;
      If Scenario[Scene_No].Shp_No=0 then No_Data:=True;
    End;
```

```
Procedure Do_Anim_Status;
  Begin
    For X:=1 to 200 do
      With Anim[X] do
        If Active then
          Begin
            Case Anm_Nb of
              1,2:Begin
                    Inc(Ctr);
If Ctr>Bank_Anim[Anm_Nb].Total then Active:=False;
                  End;
              3 :Begin
                    Inc(Yp,Vy);
                    If (Yp<0) or (Yp>198) then Active:=False;
                    Inc(Xp,Vx);
                    If (Xp<0) or (Xp>318) then Active:=False;
                  End;
            End;
          End;
  End;

Procedure Do_Ship_Status;
Var R:Real;
    Dx,Dy:Longint;
  Begin
    With Bank_Ship[1]^.Motion[1] do
      Begin
        Case Dir of
          Left: Dec(Xp,4);
          Right: Inc(Xp,4);
          Up   : Inc(Yp,4);
          Down: Dec(Yp,4);
        End;
        If Xp<0 then Xp:=0;
        If Xp>280 then Xp:=280;
        If Yp<0 then Yp:=0;
        If Yp>170 then Yp:=170;
        Inc(Ship[1].Ctr);
        If Ship[1].Ctr>5 then Ship[1].Ctr:=1;
      End;

    For X:=2 to 20 do
      With Ship[X] do
        If Active then
          Begin
            Inc(Ctr);
            If Bank_Ship[Shp_No]^.Motion[Ctr].Fire<>0 then
              Begin
                For Y:=20 to 100 do
                  With Weapon[Y] do
```

```
                    If Wpn_No=0 then
                      Begin
                        PlaySound(1);

Weapon[Y].Wpn_No:=Bank_Ship[Shp_No]^.Motion[Ship[X].Ctr].Fire;
Weapon[Y].Xp:=Bank_Ship[Shp_No]^.Motion[Ship[X].Ctr].Xp+(Picture[Bank_Ship[Shp_No]^.
Motion[Ship[X].Ctr].Pic_No].Length Div 2);

Weapon[Y].Yp:=Bank_Ship[Shp_No]^.Motion[Ship[X].Ctr].Yp+Picture[Bank_Ship[Shp_No]^.
                          Motion[Ship[X].Ctr].Pic_No].Width;
                          Weapon[Y].Ctr:=1;
                          Case Bank_Weapon[Wpn_No].Tipe of
                            3:Begin
Dx:=-Weapon[Y].Xp+Bank_Ship[1]^.Motion[1].Xp;
Dy:=-Weapon[Y].Yp+Bank_Ship[1]^.Motion[1].yp;
R:=((Sqrt(Dx*Dx+Dy*Dy)))/Bank_Weapon[Wpn_No].Vy;
                                If R=0 then R:=0.001;
                                Weapon[Y].Vx:=Round(Dx/R);
                                Weapon[Y].Vy:=Round(Dy/R);
                              End;
                          End;
                          Y:=100;
                        End;
                  End;
              End;
  End;

Procedure Explode(Crash:Boolean;Tipe:Byte;Xs,Ys:Integer);
Var Ct,Tot,X:Integer;
  Begin
    For X:=1 to 200 do
      With Anim[X] do
        If Not Active then
          Begin
            Active:=True;
            Ctr:=1;
            Anm_Nb:=Tipe;
            Xp:=Xs;Yp:=Ys;
            X:=200;
          End;
  If Tipe=2 then PlaySound(3);
  If Crash then
    Case Tipe of
      1:Begin
          PlaySound(2);
          Tot:=Round(Random(5))+3;
          Ct:=0;
          For X:=1 to 200 do
            With Anim[X] do
              If Not Active then
```

```
                Begin
                  Inc(Ct);
                  Active:=True;
                  Ctr:=1;
                  Anm_Nb:=3;
                  Xp:=Xs+Round(Random(5));Yp:=Ys;
                  Vy:=-(Round(Random(3))+2);
                  Vx:=0;
                  If Ct>=Tot then X:=200;
                End;
        End;

      2:Begin
          Tot:=Round(Random(10))+5;
          Ct:=0;
          For X:=1 to 200 do
            With Anim[X] do
              If Not Active then
                Begin
                  Inc(Ct);
                  Active:=True;
                  Ctr:=1;
                  Anm_Nb:=3;
                  Xp:=Xs+Round(Random(5));
                  Yp:=Ys+Round(Random(5));
                  Vy:=(Round(Random(12)))-6;
                  If Vy=0 then Vy:=1;
                  Vx:=(Round(Random(12)))-6;
                  If Vx=0 then Vx:=1;
                  If Ct>=Tot then X:=200;
                End;
        End;
    End;
  End;

Procedure Do_Dynamite;
Var Eng:Integer;
  Begin
    For X:=2 to 20 do
      With Ship[X] do
        If Active then
          Begin
            Eng:=Energy;
            Dec(Eng,Bank_Weapon[5].Destruct);
            If Eng<0 then Eng:=0;
            Energy:=Eng;
          End;
    DynaCtr:=1;
    Dynamite:=True;
    Pal_Ctr:=1;
```

```
    Set_All_Palette(User_Pal,63,63,0);
  End;

Procedure Do_Weapon_Status;
  Begin
    For X:=1 to 100 do
      With Weapon[X] do
        If Wpn_No<>0 then
          Begin
            Inc(Ctr);
            Case Bank_Weapon[Wpn_No].Tipe of
              0,1,4:Begin
                  Inc(Xp,Bank_Weapon[Wpn_No].Vx);
                  Inc(Yp,Bank_Weapon[wpn_No].Vy);
If (Xp+Picture[Bank_Weapon[Wpn_No].Pic_No].Length>319) or (Xp<0)
or  (Yp+Picture[Bank_Weapon[Wpn_No].Pic_No].Width>199)  or  (Yp<0)
then
                  Wpn_No:=0;
                End;
              2:Begin
                  Inc(Xp,Bank_Weapon[Wpn_No].Vx);
                  Inc(Yp,Bank_Weapon[wpn_No].Vy);
If (Xp+Picture[Bank_Weapon[Wpn_No].Pic_No].Length>319) or (Xp<0)
or (Yp+Picture[Bank_Weapon[Wpn_No].Pic_No].Width>199) or (Yp<0)
                  or (Ctr>=30) then
                    Begin
                      Wpn_No:=0;
                      Do_Dynamite;
                    End;
                End;
              3:Begin
                  Inc(Xp,Vx);
                  Inc(Yp,Vy);
If (Xp+Picture[Bank_Weapon[Wpn_No].Pic_No].Length>319) or (Xp<0)
or  (Yp+Picture[Bank_Weapon[Wpn_No].Pic_No].Width>199)  or  (Yp<0)
then
                  Wpn_No:=0;
                End;
              5:Begin
                  If Xp<Bank_Ship[1]^.Motion[1].Xp then Inc(Xp);
                  If Xp>Bank_Ship[1]^.Motion[1].Xp then Dec(Xp);
                  Inc(Yp,Bank_Weapon[Wpn_No].Vy);
If (Xp+Picture[Bank_Weapon[Wpn_No].Pic_No].Length>319) or (Xp<0)
or  (Yp+Picture[Bank_Weapon[Wpn_No].Pic_No].Width>199)  or  (Yp<0)
then
                  Wpn_No:=0;
                End;
            End;
If (Xp+Picture[Bank_Weapon[Wpn_No].Pic_No].Length>319) or (Xp<0)
```

```
or (Yp+Picture[Bank_Weapon[Wpn_No].Pic_No].Width>199) or (Yp<0)
then
            Wpn_No:=0;
          End;
  End;

Procedure Do_Dynamite_Status;
  Begin
    If Dynamite then
      Begin
        Inc(DynaCtr);
        If DynaCtr>30 then
          Begin
            DynaCtr:=1;
            Dynamite:=False;
          End Else
          Begin
            If Round(Random(2))=1 then

Explode(False,2,Round(Random(280)),Round(Random(160)));
            If Round(Random(2))=1 then

Explode(False,2,Round(Random(280)),Round(Random(160)));
            If Round(Random(2))=1 then

Explode(False,2,Round(Random(280)),Round(Random(160)));
            If DynaCtr mod 2 = 0 then
              For X:=2 to 20 do
                With Ship[X] do
                  If Active then
                    If Shp_No in [75,76,80] then else
                      If Energy>0 then
                        Dec(Energy,2);
            For X:=20 to 100 do
              Weapon[X].Wpn_No:=0;
          End;
      End;
  End;

Procedure Do_Status(Var No_Data:Boolean);
  Begin
    If not No_Data then Do_Scenario(No_Data);
    Do_Anim_Status;
    Do_Ship_Status;
    Do_Weapon_Status;
    Do_Dynamite_Status;
    Inc(Time);
    Dec(Scroll);
    If Scroll<1 then Scroll:=599;
    If Ending then
```

```
      Begin
        Inc(EndCtr);
        If EndCtr>100 then Luar:=True;
      End;
  End;

Procedure Do_Checking_Activity(Var Luar:Boolean;No_Data:Boolean);
Var Still_Work:Boolean;
  Begin
    Still_Work:=False;
    For X:=2 to 20 do
      With Ship[X] do
        If Active then
          Begin
            If Ctr>Bank_Ship[Shp_No]^.Tot_Mot then
              Begin
                Ctr:=1;
                Active:=False;
              End;
            Still_Work:=True;
          End;
     If No_Data and (Still_Work = False) then
       Begin
         Ending:=True;
       End;
  End;


Procedure Ship_Against_Ship;
Var EnLx,EnLy,MyLx,MyLy,MyX,MyY,EX,EY:Integer;
    Eng:Integer;
  Begin
    MyLx:=Picture[Bank_Ship[1]^.Motion[1].Pic_No].Length;
    MyLy:=Picture[Bank_Ship[1]^.Motion[1].Pic_No].Width;
    MyX:=Bank_Ship[1]^.Motion[1].Xp;
    MyY:=Bank_Ship[1]^.Motion[1].Yp;
    For X:=2 to 20 do
      With Ship[X] do
        If Active then
          Begin
            EX:=Bank_Ship[Shp_No]^.Motion[Ctr].Xp;
            EY:=Bank_Ship[Shp_No]^.Motion[Ctr].Yp;

EnLx:=Picture[Bank_Ship[Shp_No]^.Motion[1].Pic_No].Length;

EnLy:=Picture[Bank_Ship[Shp_No]^.Motion[1].Pic_No].Width;
            If (EX+EnLx>=MyX) and (EX<=MyX+MyLx) and
            (EY+EnLy>=MyY) and (EY<=MyY+MyLy) then
              Begin
                If Shp_No in[75,76,80] then
```

```
                    Begin
                      Ship[X].Active:=False;
                      PlaySound(4);
                      Case Shp_No of
                        75:Begin
                             Inc(Life);
                           End;
                        76:Begin
                             If Bomb<5 then Inc(Bomb)
                           End;
                        80:Begin
                             If MyGun<4 then Inc(MyGun);
                           End;
                      End;
                    End Else

                    Begin
Explode(True,1,MyX+(MyLx div 2),MyY+(MyLy div 2));
                      Dec(Ship[1].Energy,2);
                      Eng:=Ship[X].Energy;
                      Dec(Eng,5);
                      If Eng<0 then Eng:=0;
                      Ship[X].Energy:=Eng;
                    End;
                End;
            End;
  End;

Procedure Ship_against_Weapon;
Var MyWX,MyWY,MyWLX,MyWLY,EnWX,EnWY,EnWLX,EnWLY,
    EnLx,EnLy,MyLx,MyLy,MyX,MyY,EX,EY:Integer;
    Eng:Integer;

  Begin
    MyLx:=Picture[Bank_Ship[1]^.Motion[1].Pic_No].Length;
    MyLy:=Picture[Bank_Ship[1]^.Motion[1].Pic_No].Width;
    MyX:=Bank_Ship[1]^.Motion[1].Xp;
    MyY:=Bank_Ship[1]^.Motion[1].Yp;
    For X:=1 to 19 do
      With Weapon[X] do
        If (Wpn_No<>0) and (Wpn_No<>5) then

          Begin
            MyWX:=Weapon[X].Xp;
            MyWY:=Weapon[X].Yp;
MyWLx:=Picture[Bank_Weapon[Weapon[X].Wpn_No].Pic_No].Length;
MyWLy:=Picture[Bank_Weapon[Weapon[X].Wpn_No].Pic_No].Width;
            For Y:=2 to 20 do
              If (Ship[Y].Active) then
                If (Ship[Y].Shp_No in[75,76,80]) then else
```

```
                Begin
EX:=Bank_Ship[Ship[Y].Shp_No]^.Motion[Ship[Y].Ctr].Xp;
EY:=Bank_Ship[Ship[Y].Shp_No]^.Motion[Ship[Y].Ctr].Yp;
EnLx:=Picture[Bank_Ship[Ship[Y].Shp_No]^.Motion[1].Pic_No].Length;
EnLy:=Picture[Bank_Ship[Ship[Y].Shp_No]^.Motion[1].Pic_No].Width;
                  If (EX+EnLx>=MyWX) and (EX<=MyWX+MyWLx) and
                  (EY+EnLy>=MyWY) and (EY<=MyWY+MyWLy) then
                    Begin
Explode(True,1,MyWX+(MyWLx div 2),MyWY+(MyWLy div 2));
                      Eng:=Ship[Y].Energy;

Dec(Eng,Bank_Weapon[Weapon[X].Wpn_No].Destruct);
                      If Eng<0 then Eng:=0;
                      Ship[Y].Energy:=Eng;
                      Weapon[X].Wpn_No:=0;
                      Y:=20;
                    End;
                End;
          End;

    For X:=20 to 100 do
      With Weapon[X] do
        If Wpn_No<>0 then
          Begin
            EnWX:=Weapon[X].Xp;
            EnWY:=Weapon[X].Yp;

EnWLx:=Picture[Bank_Weapon[Weapon[X].Wpn_No].Pic_No].Length;
EnWLy:=Picture[Bank_Weapon[Weapon[X].Wpn_No].Pic_No].Width;
If Ship[1].Active then
Begin
If (MyX+MyLx>=EnWX) and (MyX<=EnWX+EnWLx) and
MyY+MyLy>=EnWY) and (MyY<=EnWY+EnWLy) then
Begin
Explode(True,1,EnWX+(EnWLx div 2),EnWY+(EnWLy div 2));
Eng:=Ship[1].Energy;
Dec(Eng,Bank_Weapon[Weapon[X].Wpn_No].Destruct);
If Eng<0 then Eng:=0;
Ship[1].Energy:=Eng;
Weapon[X].Wpn_No:=0;
                    End;
                End;
          End;
  End;

Procedure Do_Checking_Collision;
  Begin
    Ship_Against_Ship;
    Ship_Against_Weapon;
  End;
```

```
Procedure Refresh_MyShip;
  Begin
    Bomb:=5;
    Ship[1].Active:=True;
    Ship[1].Ctr:=1;
    Ship[1].Shp_No:=1;
    Ship[1].Energy:=Bank_Ship[1]^.Energy;
    Bank_Ship[1]^.Motion[1].Xp:=160;
    Bank_Ship[1]^.Motion[1].Yp:=170;
  End;

Procedure Do_Checking_Energy;
Var Xpos,Ypos:Integer;
  Begin
    For Y:=1 to 20 do
      With Ship[Y] do
        If Active then
          Begin
            If Energy<=0 then
              Begin
                Ship[Y].Active:=False;
                XPos:=Bank_Ship[Shp_No]^.Motion[Ctr].Xp+
(Picture[Bank_Ship[Shp_No]^.Motion[Ctr].Pic_No].Length div 2)-20;
YPos:=Bank_Ship[Shp_No]^.Motion[Ctr].Yp+
(Picture[Bank_Ship[Shp_No]^.Motion[Ctr].Pic_No].Width div 2)-20;
                Explode(True,2,Xpos,Ypos);
              End;
          End;
     If Ship[1].Active=False then
       Begin
         Dec(Life);
         If Life<0 then
           Begin
             Ending:=True;
           End Else Refresh_MyShip;
       End;
  End;

Procedure Do_Checking(Var Luar:Boolean;No_Data:Boolean);
  Begin
    Do_Checking_Activity(Luar,No_Data);
    If Ship[1].Active then Do_Checking_Collision;
    Do_Checking_Energy;
  End;

Procedure Put_Status;
  Begin
    For X:=1 to Bomb do
      PutImage(Proc^,10+X*10,190,Picture[96].Picture^);
    Block(Proc^,70,190,172,198,LightRed);
```

```
    Rectangle(Proc^,70,190,172,198,15);
If Ship[1].Energy>0 then Block(Proc^,71,191,71+Ship[1].Energy,197,14);
    For X:=1 to Life do
      PutImage(Proc^,190+X*15,185,Picture[103].Picture^);
  End;

Procedure Refresh_Data;
  Begin
   Life:=3;
    MyGun:=1;
    Time:=1;
    Scene_No:=1;
    For X:=1 to 20 do
      Begin
        FillChar(Ship[X],SizeOf(Rec_Ship_Status),0);
      End;
    For X:=1 to 100 do
      FillChar(Weapon[X],SizeOf(Rec_Weapon_Status),0);
    For X:=1 to 200 do
      FillChar(Anim[X],SizeOf(Rec_Anim_Status),0);
  End;

Procedure Refresh_Anim;
  Begin
    With Bank_Anim[1] do
      Begin
        Total:=8;
        Frame[1]:=128;
        Frame[2]:=128;
        Frame[3]:=129;
        Frame[4]:=129;
        Frame[5]:=130;
        Frame[6]:=130;
        Frame[7]:=131;
        Frame[8]:=131;
      End;
    With Bank_Anim[2] do
      Begin
        Total:=17;
        Frame[1]:=79;
        Frame[2]:=80;
        Frame[3]:=81;
        Frame[4]:=82;
        Frame[5]:=83;
        Frame[6]:=84;
        Frame[7]:=85;
        Frame[8]:=86;
        Frame[9]:=87;
        Frame[10]:=88;
        Frame[11]:=89;
```

```
        Frame[12]:=90;
        Frame[13]:=91;
        Frame[14]:=92;
        Frame[15]:=93;
        Frame[16]:=94;
        Frame[17]:=95;
      End;
  End;

Procedure Refresh_Sound;
  Begin
    With SEffect[2] do
      Begin
        Frame[1]:=100;
        Frame[2]:=110;
        Frame[3]:=120;
        Frame[4]:=130;
        Frame[5]:=150;
        Frame[6]:=130;
        Frame[7]:=80;
        Frame[8]:=50;
      End;

    With SEffect[1] do
      Begin
        Frame[1]:=80;
        Frame[2]:=70;
        Frame[3]:=50;
      End;

    With SEffect[3] do
      Begin
        For X:=1 to 50 do
          Frame[X]:=Round(Random(30))+40
      End;

    With SEffect[4] do
      For X:=0 to 24 do
        Begin
          Frame[X*5+1]:=X*100+100;
          Frame[X*5+2]:=X*100+100;
          Frame[X*5+3]:=X*100+100;
          Frame[X*5+4]:=X*100+100;
          Frame[X*5+5]:=X*100+100;
        End;
  End;

Procedure Do_Sound;
  Begin
    If SBuffer.Frame[Sound_Ctr]=0 then Nosound else
```

```
      Sound(SBuffer.Frame[Sound_Ctr]);
    Inc(Sound_Ctr);
    If Sound_Ctr>200 then
      Begin
        Sound_Ctr:=1;
        FillChar(SBuffer,SizeOf(SBuffer),0);
      End;
  End;

Procedure Intro_Prog;
Var Pal_Ctr:Integer;
    Direction:Boolean;
  Begin
    Load_Picture('Pic\Title.Pic',Proc^);
    MoveTo(Proc^,Screen^);
    Active_To_Palette(Default_Pal,10);
    Set_All_Palette(User_Pal,0,0,63);
    Pal_Ctr:=1;
    Direction:=True;
    Repeat
      Inc(Pal_Ctr);
      If Pal_Ctr>20 then
        Begin
          Direction:=Not Direction;
          Pal_Ctr:=1;
        End;
       If Direction then Metamorph_Palette(Active_Pal,User_Pal,1)
else
        Metamorph_Palette(Active_Pal,Default_Pal,1);
      Refresh_Palette;
      Delay(40);
    Until Keypressed;
    Ch:=Readkey;
    Set_All_Palette(User_Pal,0,0,0);
    Active_To_Palette(User_Pal,10);
  End;

Procedure Do_Palette;
  Begin
    If Dynamite and (Pal_Ctr<=15) then
      Begin
        Inc(Pal_Ctr);
        Metamorph_Palette(Active_Pal,User_Pal,2);
        Refresh_Palette;
      End Else
      Begin
        Metamorph_Palette(Active_Pal,Default_Pal,1);
        Refresh_Palette;
      End;
  End;
```

```
Procedure Main;
Var No_Data:Boolean;

  Begin
    No_Data:=False;
    Refresh_Data;
    Refresh_Anim;
    Refresh_MyShip;
    Refresh_Sound;
    Repeat
      Repeat
        Inc(Counter_Loop);
        If Counter_Loop>Loop_Time then
          Begin
            Put_Scroll_Page(Proc^,Scroll);
            Do_Status(No_Data);
            Do_Checking(Luar,No_Data);
            Put_Sprite;
            Put_Status;
            Do_Palette;
            MoveTo(Proc^,Screen^);
            Counter_Loop:=1;
          End;
        Do_Sound;
      Until Keypressed or Luar;
      If (Not Luar) and (Ship[1].Active) then
        Begin
          Ch:=Readkey;
          Case Ch of
            #75:Dir:=Left;
            #77:Dir:=Right;
            #72:Dir:=Down;
            #80:Dir:=Up;
            ' ':Firing(MyGun);
            #13:Firing(5);
            #27:Luar:=True;
          End;
        End;
    Until Luar;
  End;

Procedure End_Prog;
  Begin
    Set_All_Palette(User_Pal,0,0,0);
    Active_To_Palette(User_Pal,10);
  End;

Procedure Removal;
  Begin
    CloseVga256;
```

```
    NoSound;
  End;

Procedure Check_Timer;
Var Sec,Hour,MSec,Min:Word;
    T_Ctr:Integer;

  Begin
    Clrscr;
    TextBackground(1);
    TextColor(15);
Writeln('              FIRE FIGHTER 1.00
    Writeln;
    TextBackground(0);
    Textcolor(7);
    Writeln('CLOCK TICK TESTING ');
    Writeln(Frame_Per_Sec,' Frames Per Second On Request ');
    Writeln('-------------------------------');
    SetTime(0,0,0,0);
    For T_Ctr:=1 to 10000 do
    GetTime(Hour,Min,Sec,MSec);
    Loop_Time:=Round((10000/Frame_Per_Sec)/(Sec+(MSec/100)));
    Writeln;
    Writeln('Speed Has Already Detected ');
    Writeln('Total Ticks Per Frame = ',Loop_Time);
    Writeln;
    Delay(1000);
    Writeln('Detected Speed Trial');
    For X:=1 to Loop_Time do
      Begin
        Write('.');
        For Y:=1 to Loop_Time do Y:=Y;
      End;
    Sound(200);Delay(30);Nosound;
    Delay(1000);
    Writeln;
    Writeln('Press Enter To Continue');
    Ch:=Readkey;
  End;

Begin
  Check_Timer;
  Initialize;
  Intro_Prog;
  Main;
  End_Prog;
  Removal;
End.
```

Penjelasan LISTING program

Program di atas 75% mirip dengan proses preview scenario. Untuk itu kita hanya akan membahas bagian-bagian yang baru, proses yang sama dengan proses pada preview scenario tidak akan kita bahas lagi.

```
Begin
  Check_Timer;
  Initialize;
```

Procedure ini dipanggil untuk menampilkan intro dari permainan.

```
  Intro_Prog;
  Main;
```

Procedure ini dipanggil untuk menampilkan ending dari permainan.

```
  End_Prog;
  Removal;
End.

Procedure Intro_Prog;
Var Pal_Ctr:Integer;
    Direction:Boolean;
  Begin
```

Load gambar judul dari permainan ke virtual page.

```
    Load_Picture('Title.Pic',Proc^);
    MoveTo(Proc^,Screen^);
```

Ubah palette secara perlahan dari hitam ke Default_Pal.

```
    Active_To_Palette(Default_Pal,10);
    Set_All_Palette(User_Pal,0,0,63);
    Pal_Ctr:=1;
    Direction:=True;
```

Proses di bawah ini digunakan untuk menaikkan dan menurunkan intensitas warna biru secara berulang sehingga diperoleh effek yang menarik.

```
    Repeat
```

```
      Inc(Pal_Ctr);
      If Pal_Ctr>20 then
        Begin
          Direction:=Not Direction;
          Pal_Ctr:=1;
        End;
If Direction then Metamorph_Palette(Active_Pal,User_Pal,1) else
        Metamorph_Palette(Active_Pal,Default_Pal,1);
      Refresh_Palette;
      Delay(40);
    Until Keypressed;
    Ch:=Readkey;
```

Redupkan palette secara perlahan.

```
    Set_All_Palette(User_Pal,0,0,0);
    Active_To_Palette(User_Pal,10);
  End;

Procedure Main;
Var No_Data:Boolean;
  Begin
    No_Data:=False;
```

Isi nilai awal dari variabel yang akan kita pakai.

```
    Refresh_Data;
    Refresh_Anim;
    Refresh_MyShip;
    Refresh_Sound;
    Repeat
      Repeat
        Inc(Counter_Loop);
        If Counter_Loop>Loop_Time then
          Begin
            Put_Scroll_Page(Proc^,Scroll);
            Do_Status(No_Data);
            Do_Checking(Luar,No_Data);
            Put_Sprite;
            Put_Status;
```

Procedure untuk melakukan perubahan-perubahan palette. Seperti pada saat Anda menjatuhkan bomb. Palette akan diubah menjadi semakin kuning, kemudian kembali ke arah normal.

```
            Do_Palette;
            MoveTo(Proc^,Screen^);
```

```
          Counter_Loop:=1;
        End;
      Do_Sound;
    Until Keypressed or Luar;
    If (Not Luar) and (Ship[1].Active) then
      Begin
```

Deteksi tombol keyboard dan masukkan perintah yang diinstruksikan ke dalam variabel Dir.

```
        Ch:=Readkey;
        Case Ch of
          #75:Dir:=Left;
          #77:Dir:=Right;
          #72:Dir:=Down;
          #80:Dir:=Up;
```

Bila tombol Spacebar atau Enter ditekan maka lakukan proses penembakan. Parameter yang dimasukkan pada procedure ini adalah nomor senjata yang ditembakkan. Nomor 5 adalah bom sedangkan MyGun pertama kali diset 1, setelah Anda mengambil bonus power, MyGun akan ditambah dengan 1 sehingga senjata Anda menjadi semakin canggih.

```
          ' ':Firing(MyGun);
          #13:Firing(5);
          #27:Luar:=True;
        End;
      End;
  Until Luar;
End;
```

Procedure ini berguna untuk mengadakan inisialisasi nilai dari variabel yang dipergunakan oleh pesawat kita.

```
Procedure Refresh_MyShip;
Begin
```

Jumlah Bomb = 5

```
  Bomb:=5;
```

Perlu Anda ketahui, di sini pesawat Anda memesan tempat pada variabel SHIP[1]. Jadi pesawat musuh hanya dapat memesan tempat mulai nomor 2 hingga 20.

```
    Ship[1].Active:=True;
    Ship[1].Ctr:=1;
    Ship[1].Shp_No:=1;
    Ship[1].Energy:=Bank_Ship[1]^.Energy;
    Bank_Ship[1]^.Motion[1].Xp:=160;
    Bank_Ship[1]^.Motion[1].Yp:=170;
  End;

Procedure Refresh_Data;
  Begin
    Time:=1;
    Scene_No:=1;
```

Senjata awal untuk pesawat kita adalah senjata nomor 1. Jumlah nyawa = 3.

```
    MyGun:=1;
    Life:=3;
    For X:=1 to 20 do
      Begin
        FillChar(Ship[X],SizeOf(Rec_Ship_Status),0);
      End;
    For X:=1 to 100 do
      FillChar(Weapon[X],SizeOf(Rec_Weapon_Status),0);
```

Variabel Anim diisi dengan 0.

```
    For X:=1 to 200 do
      FillChar(Anim[X],SizeOf(Rec_Anim_Status),0);
  End;
```

Variabel ANIM dan BANK_ANIM tidak kita dapati pada FIRE FIGHTER MISSION EDITOR. Kegunaan dari variabel ANIM adalah sama seperti SHIP dan WEAPON yaitu sebagai tempat pemesanan. Tetapi digunakan untuk tempat pemesanan animasi. Animasi ini sebenarnya sama saja seperti pesawat atau pun senjata, tetapi sifatnya pasif, cuma sebagai hiasan. Animasi ini digunakan untuk menampilkan ledakan, percikan api dan pecahan pesawat bila kena tembak. Sedangkan Bank_Anim gunanya sama seperti Bank_Ship dan Bank_Weapon yaitu menyimpan data dari animasi. Pada Bank_Anim ini dikenal juga sistem frame seperti

pada Bank_Ship sehingga kita dapat menampilkan beberapa sprite secara berurutan yang memungkinkan kita membuat animasi ledakan. Pada awal dari procedure ini dilakukan pengisian data dari variabel Bank_Anim ini secara manual. Tidak seperti pengisian data Bank_Ship dan Bank_Weapon. Ini dilakukan karena animasi yang akan kita pakai cuma tiga macam yaitu ledakan, percikan api dan pecahan pesawat. Setelah memperhitungkan untung ruginya, diputuskan lebih baik memasukkan data animasi ini secara langsung karena tidak terlalu banyak, dibandingkan bila kita harus membuang waktu lagi untuk membuat program editornya.

```
Procedure Refresh_Anim;
Begin
    With Bank_Anim[1] do
      Begin
```

Jumlah frame = 8.

```
        Total:=8;
```

Frame [n] diisi dengan nomor sprite yang harus ditampilkan. Bank_Anim[1] berisi animasi percikan api.

```
        Frame[1]:=128;
        Frame[2]:=128;
        Frame[3]:=129;
        Frame[4]:=129;
        Frame[5]:=130;
        Frame[6]:=130;
        Frame[7]:=131;
        Frame[8]:=131;
      End;
```

Bank_Anim[2] berisi animasi ledakan.

```
    With Bank_Anim[2] do
      Begin
        Total:=17;
        Frame[1]:=79;
        Frame[2]:=80;
        Frame[3]:=81;
        Frame[4]:=82;
        Frame[5]:=83;
        Frame[6]:=84;
        Frame[7]:=85;
```

```
    Frame[8]:=86;
    Frame[9]:=87;
    Frame[10]:=88;
    Frame[11]:=89;
    Frame[12]:=90;
    Frame[13]:=91;
    Frame[14]:=92;
    Frame[15]:=93;
    Frame[16]:=94;
    Frame[17]:=95;
  End;
```

Animasi pecahan pesawat tidak perlu dideklarasikan karena teknik penampilannya agak lain dari kedua animasi di atas. Prosesnya akan kita bahas pada bagian PUT_SPRITE.

```
  End;

Procedure Refresh_Sound;

  Begin
```

Pemanfaatan Sound pada game FIRE FIGHTER. Kalau dilihat-lihat memang agak aneh karena untuk Sound effect dipergunakan sistem frame. Cara ini dipakai untuk mengatasi keterbatasan pemrograman dengan menggunakan fasilitas SOUND. Seperti yang telah kita ketahui bila kita hendak menampilkan sebuah suara selama beberapa saat, misalnya suara ledakan, gunakan SOUND, kemudian DELAY beberapa saat kemudian NOSOUND. Cara ini akan menghambat jalannya animasi karena setiap ada suara ledakan dilakukan DELAY beberapa saat. Untuk mengatasi masalah ini suara yang akan kita keluarkan akan kita masukkan ke dalam frame. Proses pengaksesan suara ini adalah sebagai berikut.

Kita sediakan dahulu sebuah variabel BUFFER bernama SBUFFER. Tipe variabel ini sama dengan tipe variabel SEFFECT[n]. Jika terjadi ledakan, maka secara langsung, apa yang ada pada variabel SEFFECT[3] dipindahkan ke dalam variabel SBUFFER. Pada setiap loop, program akan menjalankan procedure Do_Sound. Pada procedure tersebut, suara pada frame akan dibunyikan pada setiap kali melakukan proses looping. NOSOUND akan dilakukan kalau isi frame dari SBUFFER = 0. Dengan cara ini maka suara akan dihasilkan seiring dengan jalannya animasi.

```
    With SEffect[2] do
      Begin
        Frame[1]:=100;
        Frame[2]:=110;
        Frame[3]:=120;
        Frame[4]:=130;
        Frame[5]:=150;
        Frame[6]:=130;
        Frame[7]:=80;
        Frame[8]:=50;
      End;

    With SEffect[1] do
      Begin
        Frame[1]:=80;
        Frame[2]:=70;
        Frame[3]:=50;
      End;

    With SEffect[3] do
      Begin
```

Dengan teknik ini dihasilkan suara ledakan.

```
        For X:=1 to 50 do
          Frame[X]:=Round(Random(30))+40
      End;
    With SEffect[4] do
      For X:=0 to 24 do
        Begin
          Frame[X*5+1]:=X*100+100;
          Frame[X*5+2]:=X*100+100;
          Frame[X*5+3]:=X*100+100;
          Frame[X*5+4]:=X*100+100;
          Frame[X*5+5]:=X*100+100;
        End;
  End;
```

Procedure ini berfungsi untuk melakukan pemesanan senjata untuk pesawat kita.

```
Procedure Firing(Gun:Byte);
Begin
    If (Gun=5) and (Bomb<1) then else
    For Y:=1 to 19 do
      With Weapon[Y] do
        If Wpn_No=0 then
          Begin
```

Kalau senjata yang ditembakkan adalah bomb maka kurangi jumlah bom.

```
If Gun=5 then Dec(Bomb);
PlaySound(1);
```

Tulis informasi nomor senjata yang memesan tempat.

```
Weapon[Y].Wpn_No:=Gun;
```

Isi koordinat awal dari senjata dengan koordinat titik tengah dari pesawat Anda.

```
Weapon[Y].Xp:=Bank_Ship[1]^.Motion[1].Xp+(Picture[Bank_Ship[1]^.
Motion[Ship[1].Ctr].Pic_No].Length Div 2);
Weapon[Y].Yp:=Bank_Ship[1]^.Motion[1].Yp;
Weapon[Y].Ctr:=1;
```

Keluar dari loop.

```
            Y:=19;
          End;
  End;
```

Proses Object Status

Tidak banyak perubahan yang dilakukan pada procedure ini.

```
Procedure Do_Status(Var No_Data:Boolean);
Begin
    If not No_Data then Do_Scenario(No_Data);
```

Pemanggilan procedure ini dilakukan untuk menambah nomor sprite pada animasi yang sedang aktif.

```
    Do_Anim_Status;
    Do_Ship_Status;
    Do_Weapon_Status;
```

Pemanggilan procedure ini dilakukan untuk mengaktifkan ledakan-ledakan yang disebabkan oleh jatuhnya bom.

```
    Do_Dynamite_Status;
    Inc(Time);
```

```
    Dec(Scroll);
    If Scroll<1 then Scroll:=599;
```

Proses penambahan nilai variabel EndCtr ini dilakukan agar setelah semua musuh telah habis, program tidak langsung selesai, tetapi menunggu 100 kali loop baru kemudian keluar. Ending = TRUE berarti program sudah siap untuk keluar.

```
    If Ending then
      Begin
        Inc(EndCtr);
        If EndCtr>100 then Luar:=True;
      End;
  End;

Procedure Do_Anim_Status;
  Begin
    For X:=1 to 200 do
      With Anim[X] do
        If Active then
          Begin
             Case Anm_Nb of
              1,2:Begin
```

Jika nomor animasi = 1 atau 2 yaitu animasi ledakan atau percikan api, maka animasi tersebut akan dinonaktifkan kalau counter framenya telah mencapai jumlah maksimum frame.

```
                    Inc(Ctr);
If Ctr>Bank_Anim[Anm_Nb].Total then Active:=False;
                  End;
              3 :Begin
```

Jika nomor animasi = 3 yaitu animasi pecahan pesawat, maka animasi tersebut akan dinonaktifkan kalau koordinatnya telah melebihi batas layar.

```
                    Inc(Yp,Vy);
                    If (Yp<0) or (Yp>198) then Active:=False;
                    Inc(Xp,Vx);
                    If (Xp<0) or (Xp>318) then Active:=False;
                  End;
          End;
        End;
  End;
```

```
Procedure Do_Ship_Status;
Var R:Real;
    Dx,Dy:Longint;
  Begin
```

Proses pertama ialah menggerakkan pesawat dari pemain.

```
    With Bank_Ship[1]^.Motion[1] do
      Begin
        Case Dir of
          Left: Dec(Xp,4);
          Right: Inc(Xp,4);
          Up   : Inc(Yp,4);
          Down: Dec(Yp,4);
        End;
        If Xp<0 then Xp:=0;
        If Xp>280 then Xp:=280;
        If Yp<0 then Yp:=0;
        If Yp>170 then Yp:=170;
```

Dilakukan penambahan counter frame untuk membuat sprite yang berulang.

```
        Inc(Ship[1].Ctr);
        If Ship[1].Ctr>5 then Ship[1].Ctr:=1;
      End;
```

Proses berikutnya yaitu mengubah koordinat dan counter frame untuk setiap pesawat musuh yang aktif.

```
    For X:=2 to 20 do
      With Ship[X] do
        If Active then
          Begin
            Inc(Ctr);
```

Jika ada tembakan yang dilakukan maka dilakukan proses pemesanan senjata.

```
            If Bank_Ship[Shp_No]^.Motion[Ctr].Fire<>0 then
              Begin
                For Y:=20 to 100 do
                  With Weapon[Y] do
                    If Wpn_No=0 then
                      Begin
```

Panggil procedure PLAYSOUND. Di procedure ini dilakukan pemindahan data dari SEFFECT[1] ke SBUFFER sehingga sound effect nomor 1 akan dibunyikan oleh procedure DO_SOUND.

```
PlaySound(1);
```

Isi informasi nomor senjata yang ditembakkan.

```
Weapon[Y].Wpn_No:=Bank_Ship[Shp_No]^.Motion[Ship[X].Ctr].Fire;
```

Isi koordinat awal dari senjata dengan koordinat titik tengah dari pesawat yang menembakkannya.

```
Weapon[Y].Xp:=Bank_Ship[Shp_No]^.Motion[Ship[X].Ctr].Xp+(Picture[Bank_Ship[Shp_No]^.
Motion[Ship[X].Ctr].Pic_No].Length Div 2);
Weapon[Y].Yp:=Bank_Ship[Shp_No]^.Motion[Ship[X].Ctr].Yp+Picture[Bank_Ship[Shp_No]^.
                              Motion[Ship[X].Ctr].Pic_No].Width;
                              Weapon[Y].Ctr:=1;
                              Case Bank_Weapon[Wpn_No].Tipe of
                                3:Begin
```

Kalau tipe dari senjata adalah 3 maka senjata tersebut adalah senjata yang arahnya ditentukan pada saat ditembakkan yaitu ke arah pesawat kita maka dilakukan proses perhitungan kecepatan dari senjata tersebut. Caranya yaitu dengan mengurangi koordinat X dan Y dari pesawat kita dengan koordinat X dan Y senjata kemudian dibagi dengan jarak antara pesawat kita dengan senjata tersebut. Cara ini dikenal dengan cara proyeksi vektor ke komponen sumbu X dan ke komponen sumbu Y. Dengan rumus ini didapatkan kecepatan horizontal dan kecepatan vertikal dari senjata.

```
Dx:=-Weapon[Y].Xp+Bank_Ship[1]^.Motion[1].Xp;
Dy:=-Weapon[Y].Yp+Bank_Ship[1]^.Motion[1].yp;
R:=((Sqrt(Dx*Dx+Dy*Dy)))/Bank_Weapon[Wpn_No].Vy;
                                    If R=0 then R:=0.001;
                                    Weapon[Y].Vx:=Round(Dx/R);
                                    Weapon[Y].Vy:=Round(Dy/R);
                                  End;
                              End;
```

Keluar dari loop setelah melakukan pemesanan tempat untuk senjata.

```
                              Y:=100;
                            End;
```

```
            End;
          End;
  End;

Procedure Do_Weapon_Status;
  Begin
    For X:=1 to 100 do
      With Weapon[X] do
        If Wpn_No<>0 then
          Begin
```

Tambah counter dari senjata.

```
            Inc(Ctr);
            Case Bank_Weapon[Wpn_No].Tipe of
              0,1,4:Begin
```

Jika tipe senjata = 0,1 atau 4 maka tambah koordinat senjata tersebut dengan kecepatannya. Kecepatan tersebut disimpan dalam Bank_Weapon.

```
                  Inc(Xp,Bank_Weapon[Wpn_No].Vx);
                  Inc(Yp,Bank_Weapon[Wpn_No].Vy);
If (Xp+Picture[Bank_Weapon[Wpn_No].Pic_No].Length>319) or (Xp<0)
or (Yp+Picture[Bank_Weapon[Wpn_No].Pic_No].Width>199) or (Yp<0)
then
                  Wpn_No:=0;
                End;
              2:Begin
```

Jika tipe senjata adalah nomor 2 maka senjata tersebut adalah bom. Untuk bom, bila counter telah mencapai 30 maka dilakukan proses ledakan dengan dipanggilnya procedure DO_DYNAMITE.

```
                  Inc(Xp,Bank_Weapon[Wpn_No].Vx);
                  Inc(Yp,Bank_Weapon[Wpn_No].Vy);
If (Xp+Picture[Bank_Weapon[Wpn_No].Pic_No].Length>319) or (Xp<0)
or (Yp+Picture[Bank_Weapon[Wpn_No].Pic_No].Width>199) or (Yp<0)
                  or (Ctr>=30) then
                    Begin
                      Wpn_No:=0;
                      Do_Dynamite;
                    End;
                End;
              3:Begin
```

Jika Tipe = 3 maka senjata tersebut adalah senjata yang arahnya baru dihitung pada saat ditembakkan. Maka tambahkan koordinat senjata dengan kecepatan horizontal dan vertikal hasil perhitungan yaitu disimpan pada Weapon[n].Vx dan Weapon[n].Vy bukannya dari Bank_Weapon.

```
                    Inc(Xp,Vx);
                    Inc(Yp,Vy);
If (Xp+Picture[Bank_Weapon[Wpn_No].Pic_No].Length>319) or (Xp<0)
or (Yp+Picture[Bank_Weapon[Wpn_No].Pic_No].Width>199) or (Yp<0)
then
                    Wpn_No:=0;
                  End;
               5:Begin
```

Jika tipe senjata = 5 maka senjata tersebut semi-mengejar atau dengan kata lain mengejar secara horizontal. Jika senjata tersebut ada di sebelah kiri Anda maka dia akan berusaha bergerak ke kanan, demikian sebaliknya bila dia berada di sebelah kanan Anda. Tetapi arah vertikalnya selalu ke bawah layar.

```
                    If Xp<Bank_Ship[1]^.Motion[1].Xp then Inc(Xp);
                    If Xp>Bank_Ship[1]^.Motion[1].Xp then Dec(Xp);
                    Inc(Yp,Bank_Weapon[Wpn_No].Vy);
If (Xp+Picture[Bank_Weapon[Wpn_No].Pic_No].Length>319) or (Xp<0)
or (Yp+Picture[Bank_Weapon[Wpn_No].Pic_No].Width>199) or (Yp<0)
then
                    Wpn_No:=0;
                  End;
            End;
```

Jika senjata tersebut melebihi batas layar maka dia akan dinonaktifkan. Sebenarnya proses ini tidak diperlukan karena masing-masing case di atas sudah memilikinya, tapi tetap ditulis dengan alasan keamanan.

```
If (Xp+Picture[Bank_Weapon[Wpn_No].Pic_No].Length>319) or (Xp<0)
or (Yp+Picture[Bank_Weapon[Wpn_No].Pic_No].Width>199) or (Yp<0)
then
            Wpn_No:=0;
          End;
  End;
```

Procedure ini bertugas untuk menampilkan ledakan-ledakan dengan melakukan pemesanan tempat untuk animasi ledakan pada variabel

ANIM. Selain itu dilakukan juga pengurangan power dari musuh yang sedang aktif dan penonaktifan semua senjata musuh selama ledakan ini berlangsung.

```
Procedure Do_Dynamite_Status;
Begin
    If Dynamite then
      Begin
```

Tambahkan counter dari dynamite.

```
        Inc(DynaCtr);
```

Kalau counter dari dynamite telah mencapai 30 maka ledakan telah selesai.

```
        If DynaCtr>30 then
          Begin
            DynaCtr:=1;
            Dynamite:=False;
          End Else
          Begin
```

Lakukan pemanggilan procedure EXPLODE bila syarat random dipenuhi. Procedure EXPLODE ini akan melakukan pemesanan tempat untuk animasi ledakan pada variabel ANIM. Maksud dari parameter FALSE yang dikirimkan adalah jangan mengaktifkan animasi pecahan pesawat. Sedangkan parameter 2 berarti animasi diambil dari Bank_Anim[2]. Kedua parameter terakhir adalah koordinat X dan Y dari ledakan di mana dipergunakan bilangan random sehingga dihasilkan ledakan-ledakan pada tempat yang acak.

```
If Round(Random(2))=1 then
Explode(False,2,Round(Random(280)),Round(Random(160)));
If Round(Random(2))=1 then
Explode(False,2,Round(Random(280)),Round(Random(160)));
If Round(Random(2))=1 then
Explode(False,2,Round(Random(280)),Round(Random(160)));
If DynaCtr mod 2 = 0 then
```

Kurangi power semua pesawat musuh selama ledakan berlangsung. Jangan kurangi power pesawat berindeks Bank_Ship 75,76,80 karena ketiga cell Bank_Ship ini hanya berisi bonus, bukannya pesawat musuh.

```
            For X:=2 to 20 do
              With Ship[X] do
                If Active then
                  If Shp_No in [75,76,80] then else
                    If Energy>0 then
                      Dec(Energy,2);
```

Nonaktifkan semua senjata musuh. Hanya pada Weapon[20] hingga Wapon[100] karena untuk Weapon di bawah 20 digunakan untuk pemesanan senjata pada pesawat kita.

```
          For X:=20 to 100 do
            Weapon[X].Wpn_No:=0;
        End;
    End;
End;
```

Procedure ini berfungsi untuk melakukan pemesanan animasi ledakan, percikan api dan juga pecahan pesawat.

```
Procedure Explode(Crash:Boolean;Tipe:Byte;Xs,Ys:Integer);
Var Ct,Tot,X:Integer;
  Begin
```

Lakukan pemesanan animasi ledakan jika Tipe = 2, atau percikan api jika Tipe = 1.

```
  For X:=1 to 200 do
    With Anim[X] do
      If Not Active then
        Begin
          Active:=True;
          Ctr:=1;
          Anm_Nb:=Tipe;
          Xp:=Xs;Yp:=Ys;
          X:=200;
        End;
If Tipe=2 then PlaySound(3);
```

Jika Crash = True maka lakukan pemesanan animasi pecahan pesawat.

```
If Crash then
  Case Tipe of
    1:Begin
```

Jika tipe animasi adalah animasi percikan api.

```
PlaySound(2);
Tot:=Round(Random(5))+3;
Ct:=0;
For X:=1 to 200 do
  With Anim[X] do
    If Not Active then
      Begin
        Inc(Ct);
        Active:=True;
        Ctr:=1;
```

Isi Anm_Nb dengan 3 yang menunjukkan animasi pecahan pesawat. Lakukan pemesanan animasi pecahan pesawat dengan arah ke atas layar. Hal ini dapat Anda lihat apabila sebuah pesawat terkena tembakan, akan terlihat pecahan pesawat terlempar ke atas layar.

```
        Anm_Nb:=3;
        Xp:=Xs+Round(Random(5));Yp:=Ys;
        Vy:=-(Round(Random(3))+2);
        Vx:=0;
        If Ct>=Tot then X:=200;
      End;
  End;
2:Begin
```

Jika Tipe = 2 maka animasi adalah animasi ledakan.

```
Tot:=Round(Random(10))+5;
Ct:=0;
```

Lakukan pemesanan animasi pecahan pesawat dengan arah random. Hal ini dapat Anda lihat jika ada sebuah pesawat yang meledak, akan terlihat pecahannya terlempar ke segala arah. Jumlahnya ditentukan oleh variabel Tot yang dihasilkan secara random (lebih dari 5).

```
For X:=1 to 200 do
  With Anim[X] do
    If Not Active then
      Begin
        Inc(Ct);
        Active:=True;
        Ctr:=1;
        Anm_Nb:=3;
        Xp:=Xs+Round(Random(5));
```

```
                Yp:=Ys+Round(Random(5));
                Vy:=(Round(Random(12)))-6;
                If Vy=0 then Vy:=1;
                Vx:=(Round(Random(12)))-6;
                If Vx=0 then Vx:=1;
                If Ct>=Tot then X:=200;
              End;
        End;
    End;
  End;
```

Procedure ini berfungsi untuk mengaktifkan dynamite pada saat bom meledak.

```
Procedure Do_Dynamite;
Var Eng:Integer;
  Begin
```

Kurangi power dari semua musuh yang sedang aktif.

```
    For X:=2 to 20 do
      With Ship[X] do
        If Active then
          Begin
            Eng:=Energy;
            Dec(Eng,Bank_Weapon[5].Destruct);
            If Eng<0 then Eng:=0;
            Energy:=Eng;
          End;
```

Aktifkan Dynamite dengan cara mengisikan Dynamite = TRUE.

```
    DynaCtr:=1;
    Dynamite:=True;
    Pal_Ctr:=1;
```

Isikan palette tujuan dengan warna kuning.

```
    Set_All_Palette(User_Pal,63,63,0);
  End;
```

Proses Checking

Sama seperti pada program editor. Proses Checking di sini akan mendeteksi apakah sebuah pesawat telah mencapai frame terakhir sehingga harus dinonaktifkan. Di samping itu dilakukan pula

pendeteksian tabrakan antara pesawat dengan pesawat, juga pesawat dengan peluru. Terakhir dilakukan pendeteksian power yang tersisa dari tiap pesawat. Pesawat yang powernya telah mencapai 0 akan dinonaktifkan.

```
Procedure Do_Checking(Var Luar:Boolean;No_Data:Boolean);
  Begin
    Do_Checking_Activity(Luar,No_Data);
```

Dilakukan pendeteksian tabrakan.

```
    If Ship[1].Active then Do_Checking_Collision;
```

Dilakukan pendeteksian power dari tiap-tiap pesawat.

```
    Do_Checking_Energy;
  End;

Procedure Do_Checking_Activity(Var Luar:Boolean;No_Data:Boolean);
Var Still_Work:Boolean;
  Begin
    Still_Work:=False;
    For X:=2 to 20 do
      With Ship[X] do
        If Active then
          Begin
```

Jika pesawat tersebut telah mencapai frame terakhir maka akan dinonaktifkan.

```
            If Ctr>Bank_Ship[Ship_No]^.Tot_Mot then
              Begin
                Ctr:=1;
                Active:=False;
              End;
            Still_Work:=True;
          End;
```

Jika sudah tidak ada pesawat yang akan dimunculkan pada skenario dan juga sudah tidak ada pesawat yang aktif maka Ending diisi dengan TRUE yang berarti program siap untuk keluar.

```
      If No_Data and (Still_Work = False) then
        Begin
          Ending:=True;
```

```
      End;
  End;

Procedure Do_Checking_Collision;
  Begin
```

Deteksi tabrakan antara pesawat dengan pesawat.

```
    Ship_Against_Ship;
```

Deteksi tabrakan antara pesawat dengan senjata.

```
    Ship_Against_Weapon;
  End;


Procedure Ship_Against_Ship;
Var EnLx,EnLy,MyLx,MyLy,MyX,MyY,EX,EY:Integer;
    Eng:Integer;
  Begin
```

MyLx = Panjang pesawat kita.
MyLy = Lebar pesawat kita.
MyX = Koordinat X pesawat kita.
MyY = Koordinat Y pesawat kita.

```
    MyLx:=Picture[Bank_Ship[1]^.Motion[1].Pic_No].Length;
    MyLy:=Picture[Bank_Ship[1]^.Motion[1].Pic_No].Width;
    MyX:=Bank_Ship[1]^.Motion[1].Xp;
    MyY:=Bank_Ship[1]^.Motion[1].Yp;
    For X:=2 to 20 do
      With Ship[X] do
        If Active then
          Begin
```

EX = Koordinat X pesawat musuh.
EY = Koordinat Y pesawat musuh.
EnLx = Panjang pesawat musuh.
EnLy = Lebar pesawat musuh.

```
            EX:=Bank_Ship[Shp_No]^.Motion[Ctr].Xp;
            EY:=Bank_Ship[Shp_No]^.Motion[Ctr].Yp;
EnLx:=Picture[Bank_Ship[Shp_No]^.Motion[1].Pic_No].Length;
EnLy:=Picture[Bank_Ship[Shp_No]^.Motion[1].Pic_No].Width;
```

Jika terjadi tabrakan antara pesawat kita dengan pesawat musuh. Rumus di bawah ini adalah rumus untuk mendeteksi tabrakan antara dua buah benda yang dianggap berbentuk persegi panjang. Rumus ini telah kita bahas pada bab-bab sebelumnya.

```
If (EX+EnLx>=MyX) and (EX<=MyX+MyLx) and
(EY+EnLy>=MyY) and (EY<=MyY+MyLy) then
  Begin
```

Jika object yang ditabrak adalah bonus maka ....

```
If Shp_No in[75,76,80] then
  Begin
    Ship[X].Active:=False;
    PlaySound(4);
    Case Shp_No of
      75:Begin
```

Anda mendapat bonus nyawa.

```
          Inc(Life);
        End;
      76:Begin
```

Anda mendapat bonus bom.

```
          If Bomb<5 then Inc(Bomb)
        End;
      80:Begin
```

Kekuatan senjata Anda ditambah. MyGun menunjukkan nomor senjata yang kita pakai. Batas maximum senjata kita adalah senjata nomor 4.

```
          If MyGun<4 then Inc(MyGun);
        End;
    End;
  End Else
  Begin
```

Bila yang Anda tabrak adalah pesawat musuh maka panggil procedure EXPLODE untuk memesan percikan api.

```
Explode(True,1,MyX+(MyLx div 2),MyY+(MyLy div 2));
```

Kurangi power musuh dan power kita juga.

```
                    Dec(Ship[1].Energy,2);
                    Eng:=Ship[X].Energy;
                    Dec(Eng,5);
                    If Eng<0 then Eng:=0;
                    Ship[X].Energy:=Eng;
                  End;
              End;
          End;
  End;

Procedure Ship_against_Weapon;
Var MyWX,MyWY,MyWLX,MyWLY,EnWX,EnWY,EnWLX,EnWLY,
    EnLx,EnLy,MyLx,MyLy,MyX,MyY,EX,EY:Integer;
    Eng:Integer;
  Begin
```

MyLx = Panjang pesawat kita.
MyLy = Lebar pesawat kita.
MyX = Koordinat X pesawat kita.
MyY = Koordinat Y pesawat kita.

```
    MyLx:=Picture[Bank_Ship[1]^.Motion[1].Pic_No].Length;
    MyLy:=Picture[Bank_Ship[1]^.Motion[1].Pic_No].Width;
    MyX:=Bank_Ship[1]^.Motion[1].Xp;
    MyY:=Bank_Ship[1]^.Motion[1].Yp;
```

Proses pertama ialah mendeteksi peluru kita yang mengenai musuh.

```
    For X:=1 to 19 do
      With Weapon[X] do
        If (Wpn_No<>0) and (Wpn_No<>5) then
          Begin
```

MyWX = Koordinat X senjata kita.
MyWY = Koordinat Y senjata kita.
MyWLx = Panjang senjata kita.
MyWLy = Lebar senjata kita.

```
            MyWX:=Weapon[X].Xp;
            MyWY:=Weapon[X].Yp;
MyWLx:=Picture[Bank_Weapon[Weapon[X].Wpn_No].Pic_No].Length;
MyWLy:=Picture[Bank_Weapon[Weapon[X].Wpn_No].Pic_No].Width;
```

Cari pesawat musuh yang kena peluru kita. Mulai pesawat nomor 2 hingga 20.

```
For Y:=2 to 20 do
  If (Ship[Y].Active) then
    If (Ship[Y].Shp_No in[75,76,80]) then else
    Begin
```

EX = Koordinat X pesawat musuh.
EY = Koordinat Y pesawat musuh.
EnLx = Panjang pesawat musuh.
EnLy = Lebar pesawat musuh.

```
EX:=Bank_Ship[Ship[Y].Shp_No]^.Motion[Ship[Y].Ctr].Xp;
EY:=Bank_Ship[Ship[Y].Shp_No]^.Motion[Ship[Y].Ctr].Yp;
EnLx:=Picture[Bank_Ship[Ship[Y].Shp_No]^.Motion[1].Pic_No].Length;
EnLy:=Picture[Bank_Ship[Ship[Y].Shp_No]^.Motion[1].Pic_No].Width;
```

Kalau pesawat musuh terkena tambakan kita maka....

```
If (EX+EnLx>=MyWX) and (EX<=MyWX+MyWLx) and
(EY+EnLy>=MyWY) and (EY<=MyWY+MyWLy) then
  Begin
```

Pesan animasi percikan api dan pecahan pesawat dengan memanggil procedure EXPLODE dengan parameter CRASH = TRUE.

```
Explode(True,1,MyWX+(MyWLx div 2),MyWY+(MyWLy div 2));
```

Kurangi power musuh dengan daya rusak dari senjata kita.

```
Eng:=Ship[Y].Energy;
Dec(Eng,Bank_Weapon[Weapon[X].Wpn_No].Destruct);
If Eng<0 then Eng:=0;
Ship[Y].Energy:=Eng;
```

Nonaktifkan senjata yang mengenai musuh.

```
Weapon[X].Wpn_No:=0;
```

Keluar dari looping yang terdalam karena peluru tersebut telah nonaktif.

```
Y:=20;
```

```
                End;
            End;
        End;
```

Proses kedua adalah mendeteksi peluru musuh yang mengenai pesawat kita. Ini yang paling tidak enak karena pesawat kita yang kena tembak. Struktur programnya harus sama seperti di atas supaya jujur.

```
    For X:=20 to 100 do
      With Weapon[X] do
        If Wpn_No<>0 then
          Begin
```

EnWX = Koordinat X senjata musuh.
EnWY = Koordinat Y senjata musuh.
EnWLx = Panjang senjata musuh.
EnWLy = Lebar senjata musuh.

```
            EnWX:=Weapon[X].Xp;
            EnWY:=Weapon[X].Yp;
EnWLx:=Picture[Bank_Weapon[Weapon[X].Wpn_No].Pic_No].Length;
EnWLy:=Picture[Bank_Weapon[Weapon[X].Wpn_No].Pic_No].Width;
            If Ship[1].Active then
              Begin
```

Jika ada peluru musuh yang mengenai pesawat kita maka

```
                If (MyX+MyLx>=EnWX) and (MyX<=EnWX+EnWLx) and
                (MyY+MyLy>=EnWY) and (MyY<=EnWY+EnWLy) then
                  Begin
```

Pesan animasi percikan api dan pecahan pesawat dengan memanggil procedure EXPLODE dengan parameter CRASH = TRUE.

```
Explode(True,1,EnWX+(EnWLx div 2),EnWY+(EnWLy div 2));
```

Kurangi power pesawat kita dengan daya rusak dari senjata musuh.

```
                    Eng:=Ship[1].Energy;

Dec(Eng,Bank_Weapon[Weapon[X].Wpn_No].Destruct);
                    If Eng<0 then Eng:=0;
                    Ship[1].Energy:=Eng;
```

Nonaktifkan senjata tersebut.

```
                    Weapon[X].Wpn_No:=0;
                  End;
              End;
        End;
  End;

Procedure Do_Checking_Energy;
Var Xpos,Ypos:Integer;
  Begin
    For Y:=1 to 20 do
      With Ship[Y] do
        If Active then
          Begin
```

Jika power dari pesawat tersebut telah mencapai 0 maka pesawat tersebut harus dinonaktifkan.

```
            If Energy<=0 then
              Begin
                Ship[Y].Active:=False;
```

Lakukan kalkulasi untuk mencari titik tengah dari pesawat yang kemudian disimpan pada variabel Xpos dan Ypos.

```
XPos:=Bank_Ship[Shp_No]^.Motion[Ctr].Xp+
(Picture[Bank_Ship[Shp_No]^.Motion[Ctr].Pic_No].Length div 2)- 20;
YPos:=Bank_Ship[Shp_No]^.Motion[Ctr].Yp+
(Picture[Bank_Ship[Shp_No]^.Motion[Ctr].Pic_No].Width div 2)- 20;
```

Panggil procedure EXPLODE untuk menampilkan animasi ledakan pada koordinat (XPos,YPos).

```
                Explode(True,2,Xpos,Ypos);
              End;
          End;
```

Kalau pesawat nomor 1 yaitu pesawat Anda sendiri sudah tidak aktif lagi, maka kurangi nyawa yang tersisa kemudian panggil procedure REFRESH_MYSHIP untuk mengembalikan jumlah bom dan power. Jika sudah tidak ada nyawa yang tersisa maka Ending diisi dengan TRUE yang berarti program siap untuk keluar atau GAME OVER.

```
    If Ship[1].Active=False then
```

```
      Begin
        Dec(Life);
        If Life<0 then
          Begin
            Ending:=True;
          End Else Refresh_MyShip;
      End;
  End;
```

Proses Put Sprite

Proses ini adalah proses meletakkan sprite pada virtual page.

```
Procedure Put_Sprite;
Var Counter:Integer;
  Begin
```

Letakkan sprite animasi yang aktif.

```
    For X:=1 to 200 do
      With Anim[X] do
        If Active then
          Begin
              Case Anm_Nb of
```

Kalau animasi tersebut adalah ledakan atau percikan api maka tampilkan sprite sesuai dengan counter framenya.

```
                1,2:If Bank_Anim[Anm_Nb].Frame[Ctr]<>0 then
PutImage(Proc^,Xp,Yp,Picture[Bank_Anim[Anm_Nb].Frame[Ctr]].Picture^);
```

Kalau animasi tersebut adalah pecahan pesawat maka tampilkan sprite nomor random antara 132 sampai 144. Karena itulah animasi pecahan pesawat ini tidak mempergunakan variabel Bank_Anim karena gambarnya untuk setiap frame adalah random/acak.

```
3 :PutImage(Proc^,Xp,Yp,Picture[Round(Random(13)+132)].Picture^)
              End;
          End;
```

Tampilkan sprite pesawat yang aktif. Arah FOR dibalik dari 20 downto 1 agar pesawat kita (pesawat nomor 1) selalu berada paling atas (tidak ditutupi oleh sprite-sprite pesawat yang lain).

```
    For X:=20 downto 1 do
      If Ship[X].Active then
        With Bank_Ship[Ship[X].Shp_No]^.Motion[Ship[X].Ctr] do
          Begin
            If Pic_No<>0 then
              If X=1 then
```

Tampilkan sprite pesawat kita.

```
PutImage(Proc^,Bank_Ship[Ship[X].Shp_No]^.Motion[1].Xp,
Bank_Ship[Ship[X].Shp_No]^.Motion[1].Yp,Picture[Pic_No].Picture^)
              Else
```

Tampilkan sprite pesawat musuh.

```
                PutImage(Proc^,Xp,Yp,Picture[Pic_No].Picture^);
          End;
```

Tampilkan sprite senjata yang sedang aktif.

```
    For X:=1 to 100 do
      With Weapon[X] do
        If Wpn_No<>0 then
          Begin
            Counter:=Ctr Mod Bank_Weapon[Wpn_No].Tot_Rep;
            If Bank_Weapon[Wpn_No].Pic_No<>0 then
PutImage(Proc^,Xp,Yp,Picture[Bank_Weapon[Wpn_No].Pic_No+Counter].Picture^);
          End;
  End;
```

Proses Put Status

Pada program Editor, proses Put Status berfungsi untuk menampilkan nomor skenario dan timer. Pada program game engine ini Put Status berfungsi untuk menampilkan status dari pesawat Anda, yaitu jumlah bom, jumlah nyawa dan jumlah power yang tersisa.

```
Procedure Put_Status;
  Begin
```

Letakkan sprite yang jumlahnya menunjukkan jumlah bomb yang tersisa.

```
    For X:=1 to Bomb do
      PutImage(Proc^,10+X*10,190,Picture[96].Picture^);
    Block(Proc^,70,190,172,198,LightRed);
    Rectangle(Proc^,70,190,172,198,15);
```

Gambar balok berwarna kuning sepanjang jumlah power yang tersisa.

```
If Ship[1].Energy>0 then
Block(Proc^,71,191,71+Ship[1].Energy,197,14);
```

Letakkan sprite yang jumlahnya menunjukkan jumlah nyawa yang tersisa.

```
    For X:=1 to Life do
      PutImage(Proc^,190+X*15,185,Picture[103].Picture^);
  End;
```

Proses Tambahan

Untuk meramaikan dan memperbagus game, ditambahkan procedure untuk menghasilkan sound effect dan juga procedure yang memanipulasi palette.

```
Procedure Do_Sound;
  Begin
```

Bunyikan suara pada frame yang sekarang. Jika isi dari frame = 0 maka dilakukan NOSOUND.

```
    If SBuffer.Frame[Sound_Ctr]=0 then Nosound else
      Sound(SBuffer.Frame[Sound_Ctr]);
```

Tambahkan Sound Frame Counter dengan 1.

```
    Inc(Sound_Ctr);
```

Jika semua frame pada SBUFFER telah dibunyikan maka isi semua frame pada SBUFFER dengan 0 sehingga pada looping berikutnya suara yang ada pada SBUFFER ini tidak dibunyikan lagi.

```
    If Sound_Ctr>200 then
      Begin
        Sound_Ctr:=1;
        FillChar(SBuffer,SizeOf(SBuffer),0);
      End;
  End;
```

Procedure ini memegang peranan pada saat bomb meledak. Saya kira Anda telah memahami proses kerja procedure di bawah ini mengingat kita telah banyak bermain-main dengan Palette pada bab-bab terdahulu.

```
Procedure Do_Palette;
Begin
    If Dynamite and (Pal_Ctr<=15) then
      Begin
        Inc(Pal_Ctr);
```

Ubah palette yang aktif ke arah User_Pal. Perubahan ini dilakukan dengan Metamorph_Palette bukannya dengan Active_To_Palette agar perubahan palette ini dilakukan seiring dengan jalannya animasi.

```
        Metamorph_Palette(Active_Pal,User_Pal,2);
        Refresh_Palette;
      End Else
      Begin
        Metamorph_Palette(Active_Pal,Default_Pal,1);
        Refresh_Palette;
      End;
  End;
```

Procedure ini selalu dipanggil pada saat sebuah sound effect akan dibunyikan.

```
Procedure PlaySound(No:Byte);
Begin
```

Pindahkan frame-frame yang berisi frekwensi pada SEFFECT[No] ke dalam SBUFFER agar dibunyikan pada saat procedure DO_SOUND dipanggil.

```
    Move(SEffect[No],SBuffer,SizeOf(SBuffer));
    Sound_Ctr:=1;
  End;
```

INFORMASI DAFTAR FILE

LISTING PROGRAM

- ANIM1.PAS
  Program instalasi mode graphic dengan GRAPH.TPU.
- ANIM2.PAS
  Program merancang gambar rumah dengan GRAPH.TPU.
- ANIM3.PAS
  Program animasi bola berjalan.
- ANIM4.PAS
  Program permainan memukul semut.
- BAB21.PAS
  Program pembuktian kelemahan GRAPH.TPU yang pertama.
- BAB22.PAS
  Program pembuktian kelemahan GRAPH.TPU yang kedua.
- SCREEN1.PAS
  Program menampilkan titik pada mode graphic $13.
- SCREEN2.PAS
  Program menampilkan garis-garis 256 warna.
- SCREEN3.PAS
  Program permainan sederhana mengambil permata.
- PALETTE1.PAS
  Program pengaturan palette dengan interrupt.
- PALETTE2.PAS
  Program pengaturan palette dengan port.
- VGACAD.PAS
  Unit animasi dan game bermodus graphic $13.
- GEOMETRI1.PAS
  Program penggunaan VGACAD dalam merancang bangun geometri.
- 3SCR1.PAS
  Program animasi sprite kursor yang dapat digerakkan.
- 3SCR2.PAS
  Program penyimpanan sprite kursor ke dalam file.
- 3SCR3.PAS
  Program pembacaan sprite kursor dari file untuk digerakkan di layar.
- CAP2.PAS
  Program capture untuk menangkap gambar di layar monitor.

- PAL1.PAS
  Program animasi palette dengan menggunakan teknik brightness.
- PAL2.PAS
  Program animasi palette dengan menggunakan teknik RGB contrast.
- PAL3.PAS
  Program animasi palette dengan menggunakan teknik gabungan (TV).
- JOIN.PAS
  Program untuk menggabung empat file PIC menjadi satu file besar untuk proses scrolling.
- SCROLL1.PAS
  Program animasi scroll dengan VGACAD.TPU.
- SCROLL2.PAS
  Program animasi scroll yang dapat Anda atur dengan kursor.
- EDITOR.PAS
  Program editor dari game FIRE FIGHTER.
- FIRE.PAS
  Program game FIRE FIGHTER.

FILE-FILE TAMBAHAN

- SMALL.OBJ
  Font dari VGAFONT.TPU.
- SCRIPT_1.OBJ
  Font dari VGAFONT.TPU.
- SCRIPT_2.OBJ
  Font dari VGAFONT.TPU.
- COMPUTER.OBJ
  Font dari VGAFONT.TPU.
- VGAFONT.TPU
  Unit aplikasi font pada modus graphic $13.
- GCAP10.EXE
  Program capture untuk menangkap gambar-gambar dengan ukuran yang dapat Anda tentukan. Hasil capture dari gambar-gambar tersebut dapat digabung dalam satu file sekaligus.
- EGAVGA.BGI
  Borland Graphic Interface file yang berguna pada saat Anda menjalankan program yang menggunakan GRAPH.TPU.

PENUTUP

Akhirnya Anda telah menguasai teknik pemrograman game FIRE FIGHTER ini. Memang agak membingungkan, tetapi memang begitulah teknik pemrograman game.

Pembahasan kita tentang pemrograman game profesional telah berakhir sampai di sini. Pada kesempatan mendatang kita kan membahas pembuatan game-game yang lebih canggih lagi seperti pembuatan game bertipe WOLFENSTEIN, WING COMMANDER, STREET FIGHTER dan lain sebagainya. Untuk itu Anda harus telah benar-benar memahami isi dari buku ini karena buku ini adalah juga merupakan dasar dari teknik pemrograman game yang disebutkan di atas.

Saya ucapkan terima kasih atas perhatian para pembaca dan saya ucapkan SEMOGA SUKSES !

Agustinus Nalwan
Monash Clayton University
Victoria-Melbourne
Australia

SERI APLIKASI PEMROGRAMAN

Pemrograman Animasi dan Game Profesional

Buku ini merupakan dasar pemrograman animasi dan game profesional. Dengan buku ini Anda akan diajak untuk mempelajari dasar-dasar pembuatan game profesional yang beredar di pasaran menggunakan compiler Turbo Pascal 7.00 yang dibantu oleh routine-routine Assembler.

Pada bab-bab awal akan dibahas tentang penggunaan GRAPH.TPU dari Turbo Pascal untuk memperkenalkan Anda tentang animasi. Selanjutnya kita akan merakit unit Graphic sendiri yaitu VGACAD.TPU, dengan unit ini Anda telah dapat merakit program game Anda sendiri yang setaraf dengan game-game yang beredar di pasaran seperti RAPTOR, STREET FIGHTER, dan sebagainya. Pada bab terakhir, kita akan mempraktekkan apa yang telah kita pelajari dengan membuat game bertipe RAPTOR.

Penerbit PT Elex Media Komputindo
Jl. Palmerah Selatan 22
Jakarta 10270

Kelompok
Pemrograman
Ketrampilan
☑ Tingkat Pemula
☑ Tingkat Menengah
☐ Tingkat Mahir
Jenis Buku
☐ Referensi
☑ Tutorial
☐ Latihan
ISBN 979-637-340-8
12195321